LNCS 3744

Thomas Magedanz
Ahmed Karmouch
Samuel Pierre
Iakovos Venieris (Eds.)

Mobility Aware Technologies and Applications

Second International Workshop, MATA 2005
Montreal, Canada, October 2005
Proceedings

Springer

Thomas Magedanz Ahmed Karmouch
Samuel Pierre Iakovos Venieris (Eds.)

Mobility Aware Technologies and Applications

Second International Workshop, MATA 2005
Montreal, Canada, October 17-19, 2005
Proceedings

Springer

Volume Editors

Thomas Magedanz
Technical University of Berlin/Fraunhofer Institute FOKUS
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
E-mail: magedanz@fokus.fraunhofer.de

Ahmed Karmouch
University of Ottawa
School of Information Technology and Engineering
Ottawa, Ontario, K1N 6N5, Canada
E-mail: karmouch@site.uottawa.ca

Samuel Pierre
École Polytechnique de Montréal
C.P. 6079, Station Centre-Ville, Montréal, H3C 3A7, Québec, Canada
E-mail: Samuel.Pierre@larim.polymtl.ca

Iakovos Venieris
National Technical University of Athens
Department of Electrical and Computer Engineering
9 Heroon Polultechneiou str., 15773 Athens, Greece
E-mail: venieris@cs.ntua.gr

Library of Congress Control Number: 2005933615

CR Subject Classification (1998): C.2, H.4, H.5, H.3

ISSN	0302-9743
ISBN-10	3-540-29410-4 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-29410-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11569510 06/3142 5 4 3 2 1 0

Preface

The beginning of the twenty-first century is characterized by global markets, and the mobility of people is becoming an important fact of life. Consequently, the mobile user is demanding appropriate technical solutions to make use of customized information and communication services.

In this context the notion of next-generation networks (NGNs), which are driven by the convergence of the entertainment sector, the mobile Internet, and fixed/mobile telecommunications, is emerging. Such NGNs are aggregating a variety of different access networks and supporting the seamless connection of an open set of end-user devices, and due to the adoption of an all-IP network paradigm they enable a much better integration of voice and data services. Coincidently the buzzword 'fixed mobile convergence' (FMC) describes the current trend towards providing common services across fixed and mobile networks resulting in the medium term in the full integration of fixed and mobile telecommunication networks.

The adoption of appropriate middleware technologies and the provision of so-called service delivery platforms driven by the ongoing innovation in the field of information technologies provides today the technical foundation for supporting terminal, personal and service mobility and thus the implementation of real seamless information and communication services. Furthermore, users are nowadays looking, in light of an omnipresent service environment, for a much higher degree of customization and context awareness in the services they use.

The papers in this volume look at these enabling mobility-aware technologies and their use for implementing mobility-aware and context-aware applications. Particularly they look at NGN service delivery platform technologies and service control issues for multimedia services, mobility-aware middleware and mobile agent platform technologies, solutions for wireless networks, mobility tracking technologies, new mobile applications and intelligent appliances, context-aware and policy-based and autonomic communications, ambient networks and smart end systems, issues and applications for mobile ad hoc networks and sensor networks, and security technologies for NGNs and context-aware environments.

We would like to express our appreciation and thanks to the authors for their contributions as well as the technical Program Committee and the reviewers who helped to put together an excellent technical program for the workshop. Special thanks are due to Edmundo Madeira and Ahmed Karmouch, who kindly contributed their time and effort to help with the organization of the review process and the technical program.

August 2005 Thomas Magedanz

Organization

General Co-chairs

Ahmed Karmouch, University of Ottawa, Canada
Samuel Pierre, École Polytechnique de Montréal, Canada

Program Co-chairs

Thomas Magedanz, Technical University of Berlin/Fraunhofer FOKUS, Germany
Iakovos Venieri, National Technical University of Athens, Greece

Tutorial Chair

Steven Chamberland, École Polytechnique de Montréal, Canada

Publicity Co-chairs

Ronald Beaubrun, Université Laval, Canada
José Fernandez, École Polytechnique de Montréal, Canada

Local Arrangements Co-chairs

Betty Monplaisir, École Polytechnique de Montréal, Canada
Raymond Lévesque, Bureau des Congrès Universitaires, Canada

Steering Committee

Ahmed Karmouch, University of Ottawa, Canada
Thomas Magedanz, Technical University of Berlin/Fraunhofer FOKUS, Germany
Roch Glitho, Ericsson Canada, Canada

IEEE ComSoc Liaison

Algirdas Pakstas, London Metropolitan University, UK

In Cooperation with

IEEE Computer Society
IFIP

Program Committee

T. Araragi, NTT, Japan
S. Arbanowski, FOKUS, Germany
J.-L. Bakker, Telcordia, USA
L.O. Barbosa, UCM, Spain
W. Bauerfeld, T-Systems, Germany
P. Bellavista, Bologna, Italy
F. Bellifemine, TILab, Italy
R. Boutaba, Univ. of Waterloo, Canada
P. Brezillon, LIP6, France
B. Burg, Panasonic, USA
M. Chen, III, Taiwan
S. Cherkaoui, Université de Sherbrooke, Canada
J. Celestino Júnior, FUC, Brazil
J. Delgado, UPF Barcelona, Spain
M. Devetsikiotis, NCSU, USA
P. Dini, Cisco, USA
J.M.H. Elmirghani, UWS, UK
M. Eriksson, Luleå Univ., Sweden
B. Falchuk, Telcordia, USA
P. Feil, Deutsche Telekom, Germany
A. Galis, UCL, UK
M.-F. Gervais, LIP6, France
R. Glitho, Ericsson, Canada
R. Giaffreda, BT, UK
S. Guan, NUS, Singapore
H. Hanarran, University of the Witwatersrand, South Africa
S. Honiden, NII, Japan
E. Horlait, LIP6, France
A. Iwasaki, NTT Labs, Japan
C. Jedrzejek, PUT, Poznan, Poland

Table of Contents

Web Agent Supporting Transport Layer Mobility

Yong-Jin Lee[1] and M. Atiquzzaman[2]

[1] Department of Technology Education,
Korea National University of Education, 363-791
yjlee1026@daum.net
[2] School of Computer Science, University of Oklahoma,
200 Felgar Street, Norman, OK 73019, USA
atiq@ou.edu

Abstract. Typical transmission control protocol (TCP) based web agents in mobile wireless environment have several deficiencies, such as performance degradation, head-of-line blocking, and unsupported mobility. The Stream Control Transmission Protocol (SCTP) is a new transport protocol, which provides multi-streaming and multi-homing features. SCTP extensions for dynamic address reconfiguration will support transport layer mobility. We address SCTP based web agents supporting seamless transport layer mobility in the wired and wireless environment. Our proposed SCTP based mobile web agent is composed of application engine to use the hypertext transfer protocol (HTTP) and protocol engine to deploy SCTP with dynamic address reconfiguration. We have investigated and described the components necessary to implement mobile web agents in a ubiquitous environment. Mean response time is an important performance measure of web agents. Simulation results show that our SCTP based web agent reduces the mean response time of a typical TCP based web agent by 12 % on the average.

1 Introduction

Ubiquitous computing environment is composed of cheap and small computers connected to the Internet, and provides customized services according to the needs of users. A network architecture supporting ubiquitous computing environment is expected to have the IPv6 backbone network and an attached IPv4 Internet that connects mobile and wireless networks, i.e. it is based on the all-IP network which provides wired and wireless services while supporting host and user mobility.

A web agent, which we address in this paper, works in the ubiquitous environment and provides mobile users with customized web services to use hypertext transfer protocol (HTTP) automatically without user intervention. For example, portable web agents installed in personal digital assistants (PDA) or notebook computers of roaming users can download objects from web servers based on a predefined profile. Since the vast majority of IP traffic is transmitted using TCP, typical web agents use Transmission Control Protocol (TCP) Application Programming Interface (API). However, TCP-based mobile web agents suffer from the following three deficiencies: performance degradation, Head-of-Line (HOL) blocking, and unsupported mobility [1,2].

Stream Control Transmission Protocol (SCTP) [3] has been proposed by IETF to overcome the above deficiencies of TCP. The performance degradation problem is alleviated in SCTP by incorporating the enhanced features of TCP congestion control schemes. For example, SCTP uses the Selective Acknowledgement (SACK)-based Fast Retransmit algorithm used in TCP. This scheme speeds up loss detection and increases bandwidth utilization [4]. Use of SACK is mandatory in SCTP. In addition, SCTP can improve throughput by increasing the congestion window size (*cwnd*) only when the full *cwnd* is utilized [5].

To overcome the HOL blocking problem of TCP, we use SCTP's multi-streaming feature to speed up the transfer of web objects. By transmitting each object in a separate stream, the HOL effect between different objects is eliminated. If one object is lost during the transfer, others can be delivered to the web agent at the upper layer while the lost object is being retransmitted from the web server. This results in a better response time to users with only one SCTP association for a particular HTML page.

Finally, to deal with the unsupported mobility problem of TCP, we utilize the extended SCTP multi-homing feature. SCTP multi-homing allows a single SCTP endpoint to support multiple IP addresses. In its current form, SCTP multi-homing support is only for redundancy. Recently, load-sharing SCTP (LS-SCTP) [6] has been suggested to aggregate the bandwidth of all active transmission paths between communicating endpoints. The extended SCTP multi-homing feature (called dynamic IP address reconfiguration [7]) is capable of supporting transport layer mobility. This feature provides a mechanism that allows an SCTP endpoint to dynamically add and delete IP addresses during the lifetime of an SCTP association. Mobile SCTP [8] and Seamless IP-diversity based Generalized Mobility Architecture (SIGMA) [9] utilize the dynamic IP address reconfiguration feature of SCTP, without requiring any modification to the IP infrastructure. Secure SCTP [10] deals with the traffic redirection attack problem that can arise from dynamic IP address reconfiguration.

While SCTP can solve several deficiencies of TCP for mobility in the ubiquitous environment, it does not provide the location management function that Mobile IP supports intrinsically. Hence, location management in SCTP is performed by the Domain Name Server (DNS) in the application layer [9] or Mobile IP in the network layer [8]. Use of DNS can cause scalability problem, and the scheme used in mobile IP can result in complexity and inefficiency in the network. In this paper, we consider a web agent that always initiates the connection setup to web servers. Location management is, therefore, outside the scope of this paper.

We propose a web agent based on SCTP and its dynamic IP address reconfiguration feature. The most important performance measure in a web environment is the mean response time between HTTP requests and replies. To compare the performance of our proposed SCTP based web agent and a typical TCP based web agent, we have carried out simulation on our experimental testbed. Results show that our web agent's mean response time is less than a TCP based web agent by about 12% on average.

The main contributions of this paper are: (i) propose an SCTP based mobile web agent architecture; (ii) describe the functions of the components of the web agent; and (iii) evaluate the performance of the proposed web agent and compare with a typical TCP-based web agent.

The rest of the paper is organized as follows. Section 2 describes the architecture and functions of the proposed SCTP based web agent. Section 3 presents results of performance evaluation, and Section 4 concludes the paper.

2 Architecture of the SCTP-Based Mobile Web Agent

We present the architecture of the SCTP-based mobile web agent in Fig. 1. It consists of an *application engine* performing the function of user interface and an SCTP-based *protocol engine* supporting transport mobility. An SCTP API serves as the interface between the two engines. The application engine exchanges HTTP message with the web server in the application layer. The protocol engine, located in the transport layer, has the mobility support component and the data component. The functions of the mobility support component are movement detection and handover management. The data component includes Transmission Control Block (TCB) and Common Data Block (CDB), which maintain related information regarding the current association and mobility support.

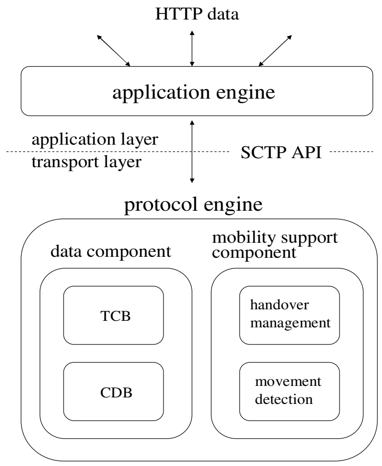
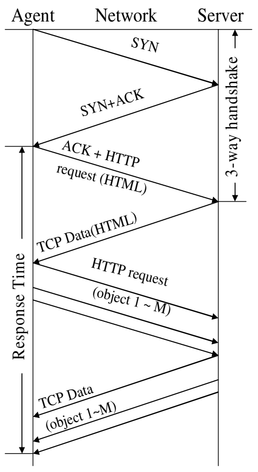


Fig. 1. Architecture of our proposed SCTP based mobile web agent

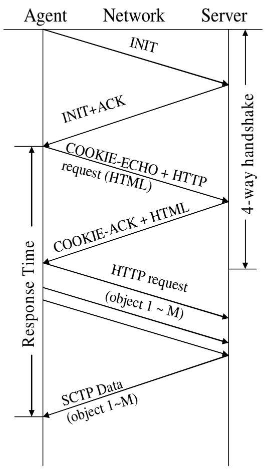
2.1 Application Engine

The application engine provides users with HTTP service, such as web content delivery. Since the application engine interacts with SCTP in the transport layer, it avoids HOL blocking (which TCP based applications experience) and reduces the response time of an application. A typical web document contains several multimedia objects, such as image, voice and text. If any object is lost during transmission, the receiver does not display the previously received object until the lost object has been completely received. This HOL blocking increases the response time, especially in wireless networks with high packet loss. Our application engine can display objects more promptly by using the multi-streaming feature which evades the HOL blocking.

TCP web agent receives an HTML file with M embedded objects after 3-way-handshake using HTTP (it hereafter refers to HTTP 1.1 with persistent connection and pipelining its requests: Fig.2 (a)). That is, TCP web agent sends M requests simultaneously using pipelining. A server sends its responses to those requests in the same order that the requests were received.



(a) TCP web agent



(b) SCTP web agent

Fig. 2. Timelines for TCP web agent and SCTP web agent

The initialization of an association in SCTP web agent using HTTP (Fig.2 (b)) is completed after the exchange of four messages. The passive side of the association does not allocate resources for the association until the third of these messages has arrived and been validated. Last two messages of the four-way handshake can carry user data. With this piggybacking, SCTP web agent has the same connection-establishment delay as TCP web agent, namely one round trip time. Furthermore, SCTP web agent does not limit the maximum number of objects for pipelining. To summarize, the main difference between TCP web agent and SCTP web agent is that SCTP web agent can receive multiple objects in parallel using multi-stream.

Furthermore, SCTP web agent has the following additional advantages when compared with a TCP web agent: (i) The security of an application is increased remarkably. Since TCP web agent establishes the connection setup using 3-way-handshake, it is defenseless against blind SYN attacks. On the other hand, our SCTP based engine strengthens the security using the signed state-cookie in the 4-way-handshake for the connection setup. Although it uses four packets for connection establishment, it can combine the HTTP request into the third packet. Consequently, there is no extra overhead compared with the TCP web agent. (ii) The fault tolerance of an application is enhanced. TCP web agent uses only one connection path. If the TCP path is broken due to the physical layer problems, data cannot be transferred. However, our engine can continue to communicate using the alternate path which multi-homing feature provides. (iii) Seamless mobility for an application is supported. An existing TCP application suffers from the disconnection because it cannot change the currently bound IP address in TCB into the new IP address. However, our web agent continues to maintain the seamless connection with web server because it can modify the TCB using the dynamic address reconfiguration of SCTP.

SCTP API helps the application programmer who is familiar with TCP and UDP to quickly adapt to SCTP. By using the API, we can easily transform a typical TCP application into the SCTP application. For example, while the TCP application uses *socket*(*AF_INET, SOCK_STREAM, IPPROTO_TCP*) as the socket system call, SCTP application uses *socket*(*AF_INET, SOCK_STREAM, IPPROTO_SCTP*). System calls, *bind*(), *sctp_bindx*() are used for address binding. Several predefined events such as the new address acquisition are delivered to the application engine via SCTP notification. Application engine can obtain the status information such as contents and addresses of event by using *sctp_opt_info*(). To enhance flexibility, an implementation including the linux kernel stream control transmission protocol (lksctp) [11] should provide an SCTP API to enable programmers to specify some parameters peculiar to SCTP, such as the number of outgoing streams to set up during negotiation, stream IDs used, etc.

2.2 Protocol Engine

In this section, we describe in detail the various components of our proposed mobile web agent.

2.2.1 Data Component

The data component defines necessary parameters for protocol implementation. CDB is necessary for SCTP instance as follows: (i) Data consumer list related with the currently connected association. Data consumer means process identification information, such as file descriptor, pipe pointer, and table pointer; (ii) Secret key for the security of end-user; (iii) Address list indicating end points; (iv) Port number indicating the binding port number of end point.

Some important parameters are stored in TCB per association as follows: (i) Peer verification tag indicating the authentication value of the correspondent node; (ii) My verification tag indicating the authentication value of local node; (iii) State indicating the current status of SCTP such as the connection complete, shutdown, and stop; (iv) Peer transport address list indicating the transport address list of the correspondent node; (v) Local transport address list indicating the local IP address list; (vi) Primary path indicating the primary destination address of the correspondent node.

2.2.2 Mobility Support Component

This component deals with the seamless transport mobility of web agent using the SCTP address configuration change (*ASCONF*) extension [7]. ASCONF extension defines the new IP address insertion (*add_ip_address*), the old IP address deletion (*delete_ip_address*), and primary IP address change (*set_primary_address*) chunks. According to the receipt of the above chunks, protocol engine changes the peer transport address list, local transport address list, and primary address stored in the TCB dynamically.

Generally, the mobility support problem in the wireless mobile network includes movement detection, handover management, and location management. Movement detection is a function of the mobile node (MN) to identify and trace its own location change. Location management is a function of the correspondent node (CN) to trace the current location of MN in order to initiate the connection establishment with MN. Handover management is a function of both MN and CN to provide the roaming MN with the seamless handover.

In this paper, we consider the web environment, where MN (web agent) always initiates connection setup to CN (web server). Thus, CN does not need location management. Nevertheless, if the location management function is necessary, we can add it into the mobility support component in Fig. 1. Timeline of the mobility support component is depicted in Fig. 3.

(1) Movement detection

Initially, mobile web agent hears router advertisement (RA) from old access router (AR), and finds the network prefix included in RA (1, Fig. 3). Web agent acquires its own IP address by using stateless auto-configuration of IPv6 based on the network prefix (when using IPv6) or querying the dynamic host configuration protocol (DHCPv4/v6) server (when using IPv4/IPv6). Web agent and web server establish the association by exchanging IP addresses. At this time, each end point specifies the primary IP address on which data is sent (2, Fig. 3). Information related to the association is recorded in each TCB of web agent and web server, followed by exchange of data.

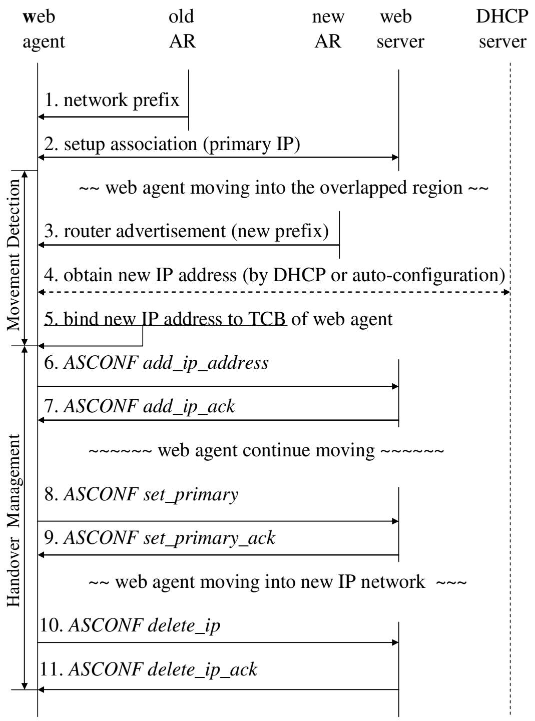

web agent
old AR
new AR
web server
DHCP server
1. network prefix
2. setup association (primary IP)
~~ web agent moving into the overlapped region ~~
3. router advertisement (new prefix)
4. obtain new IP address (by DHCP or auto-configuration)
5. bind new IP address to TCB of web agent
6. ASCONF add_ip_address
7. ASCONF add_ip_ack
~~~~~~ web agent continue moving ~~~~~~
8. ASCONF set_primary
9. ASCONF set_primary_ack
~~ web agent moving into new IP network ~~~
10. ASCONF delete_ip
11. ASCONF delete_ip_ack
Movement Detection
Handover Management

Fig. 3. Timeline for the mobility support component of web agent

The web agent, while communicating with the web server, moves from the coverage of old AR to the overlapped region which is covered by both the old and new AR's. Web agent hears new router prefix from new AR (3, Fig. 3), and detects its
~~~~~~

movement into new network by comparing its current network prefix (1, Fig. 3) with new network prefix (3, Fig. 3). If web agent uses IPv6, it can itself configure a new IP address using stateless auto-configuration based on the network prefix. Alternatively, it can acquire a new IP address from the DHCPv4/v6 server (4, Fig. 3), which increases the required signaling time. Anyway, newly obtained IP address is bound to the local transport address list in the TCB of web agent (5, Fig. 3). These events are delivered to the application engine via SCTP notification.

(2) Handover management

After binding the new IP address on TCB, web agent informs the web server that it will use the new IP address by sending *ASCONF add_ip_address* (6, Fig. 3). Web server modifies its own TCB by adding the received new IP address of web agent and replies to the web agent by an *ASCONF add_ip_ack* (7, Fig. 3). At this time, web agent becomes multi-homed, and is thus reachable by two different networks. It can receive data on both old and new IP addresses. Consequently, if there is a physical problem with the path related to the primary address, the new IP address can be used as an alternate address.

As the web agent leaves the overlapped region and enters the coverage of new AR, it experiences more packet loss on the primary path. If the amount of received packets on new IP address is greater than on the primary IP address, web agent sends out the *ASCONF set_primary* chuck to web server. This informs the web server to use the new IP address as the primary address for data communications (8, Fig. 3). Web server replies by sending an *ASCONF set_primary_ack* chunk to the web agent (9, Fig. 3).

As the web agent continues to move into the core coverage of new AR, the previous primary IP address becomes obsolete. Web agent sends out an *ASCONF delete_IP* chuck to the web server, which eliminates the previous primary IP address (10, Fig. 3). The reason to delete the obsolete IP address is as the follows: We assume that the new set primary path is broken in the coverage of new AR. If the previous primary IP address was not deleted from the binding list, it might become an alternate path. Thus, data from the web server may be redirected to the alternate path. However, the previous primary IP address cannot receive any data in the coverage of the new AR. This results in unnecessary traffic in the network. Handover is completed when the web server responds by sending *ASCONF delete_ip_ack* to the web agent (11, Fig. 3).

3 Performance Evaluation

3.1 Experimental Setup

In this section, we compare the performance of SCTP web agent with TCP web agent in terms of the mean response time. Fig. 4 shows our testbed for collecting the data. Table 1 shows the host and network configurations of the testbed.

For this experiment, we wrote two Linux C server programs that mimic HTTP over TCP and SCTP servers, respectively. We also wrote two Linux C web agent programs to simulate pipelining (TCP/SCTP) and multi-streaming (SCTP), respectively. The main reason for writing our own HTTP server and agent is due to the lack of adequate support of SCTP by current HTTP client/server.

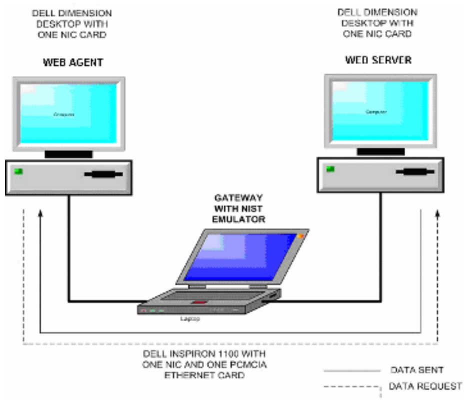

DELL DIMENSION DESKTOP WITH ONE NIC CARD
WEB AGENT
DELL DIMENSION DESKTOP WITH ONE NIC CARD
WEB SERVER
GATEWAY WITH NIST EMULATOR
DELL INSPIRON 1100 WITH ONE NIC AND ONE PCMCIA ETHERNET CARD
DATA SENT
DATA REQUEST

Fig. 4. Testbed for the simulation

Table 1. Host and network configuration of testbed

node	*Hardware*	*software*	*operating system*	*Network*
web server	Dell dimension desktop with one NIC card	TCP / SCTP server program	Redhat Linux 9 Kernel 2.6.2	eth0: 10.1.8.14, gateway: 0.1.8.5
web agent	Dell dimension desktop with one NIC card	TCP / SCTP agent program	Redhat Linux 9 Kernel 2.6.2	eth0: 10.1.10.6, gateway: 0.1.10.5
NIST emulator	Dell Inspiron-1100 laptop, one NIC card, one pcmcia Ethernet card	NIST emulator	Redhat Linux 9 Kernel 2.6.2	eth0: 10.1.8.5, gateway: default eth0: 10.1.10.5, gateway: default

In order to mimic the pipelining of TCP web agent, we used *M* threads that send requests simultaneously. TCP web agent receives HTTP reply sequentially from the server. The procedure for SCTP web agent is as follows: we first fork two processes: parent and child. In the parent process, *M* threads simulate the pipelining for sending *M* HTTP requests to the web server. In the child process, *M* threads simulate multi-streaming for receiving *M* HTTP replies from the web server. To summarize, the main difference between the TCP and SCTP web agents is that SCTP web agent can receive multiple objects in parallel using multi-streaming.

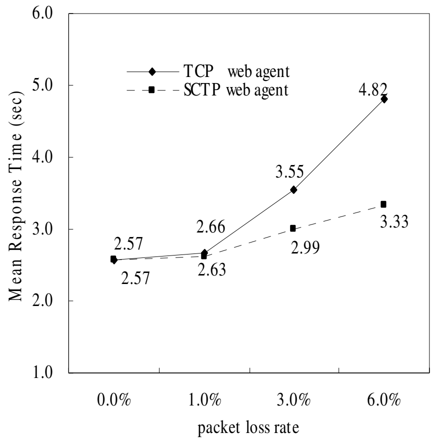
We used the NIST Net emulator [12], between the web agent and web server, to compare the performance under varying packet loss rates. We measured the response time using Ethereal protocol analyzer [13] that captures packets flowing between the server and agent. Table 2 represents test parameters used for our experiment. We have carried out the experiment using the following common parameters for all scenarios: the number of objects embedded in the HTML file (*M*) = 5, size of one object = 13. 5 KB, and maximum segment size (*MSS*) = 1,500 B. In scenario 1 of Table 2, we investigate the mean response time at packet loss rates (*p*) of 0.4 %, 1 %, 2 %, and 5 %. Bandwidth (*bw*) and round trip time (*RTT*) were set at 40 Kbps and 256 ms, respectively. In scenarios 2 and 3, we varied the bandwidth and round trip time, respectively.

Table 2. Test parameters for the experiment

Test parameters	*Scenario 1*	*Scenario 2*	*Scenario 3*
Packet loss rate (*p*: %)	0.4, 1, 2, 5	1	1
Bandwidth (*bw: bps*)	40 K	40 K, 400 K, 3 M, 10 M	40 K
round trip time (*RTT: ms*)	256	256	55, 80, 256, 1000

3.2 Experiment Result

Figs. 5~7 show the mean response times corresponding to the scenarios in Table 2. Mean response times are computed for 50 trials for each test parameter for TCP web agent and SCTP web agent, respectively.



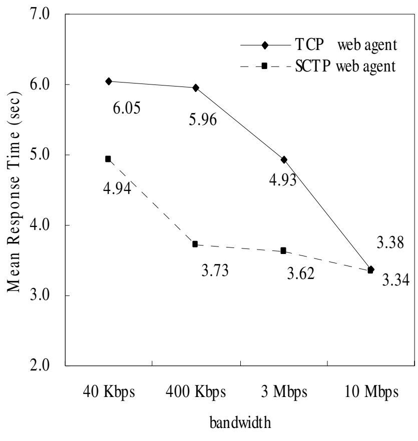


Fig. 5. Mean response time as a function of packet loss

Fig. 6. Mean response time as a function of bandwidth

Fig. 5 represents the mean response time as a function of packet loss rate. When the packet loss rate is zero (*p* = 0 %), there is no HOL blocking in a TCP web agent. Therefore, there is no difference in the mean response time between SCTP and TCP web agents. In addition, we found that the gap between TCP and SCTP web agents increases as the packet loss rate increases. This is due to the multi-streaming feature and the effective error recovery of SCTP. As the packet loss rate increases, the TCP web agent must retransmit the lost packets more frequently; moreover, it suffers from the head-of-line blocking causing the large response time. On the other hand, SCTP web agent does not suffer from the HOL blocking due to the use of the multi-streaming feature. Furthermore, the proposed SCTP web agent can detect losses more quickly than TCP web agent. Thus, it can perform fast recovery from the error using the SACK mechanism.

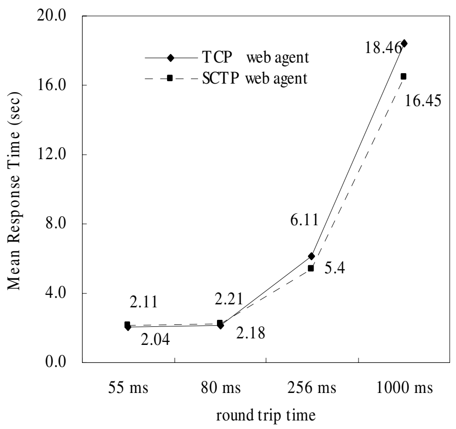
Fig. 6 shows that the mean response time is inversely proportional to the bandwidth. It is natural that larger bandwidth reduces the total transfer time, resulting in a smaller mean response time. As the bandwidth increases, the benefit of SCTP web agent over TCP web agent becomes smaller. This is because of the minimal effect of fast retransmit for high bandwidth.

In Fig. 7, we found that the rate of increase of the mean response time in TCP web agent is about 900 % between 0.055 s and 1s, while it is 800 % for SCTP web agent. Since TCP web agent cannot utilize the multi-streaming feature, all objects experience large *RTT*. On the other hand, multiple objects in SCTP web agent experience only one *RTT*. Thus, as the number of objects and *RTT* increase, the performance gap between SCTP web agent and TCP web agent increases.

We define the savings rate of SCTP web agent over TCP web agent by Eq. (1),

$$Savings\ Rate = (T_{TCP} - T_{SCTP}) \,/\, T_{TCP} \times 100\ (\%) \tag{1}$$

T_{TCP} and T_{SCTP} represent the mean response time of TCP and SCTP-based web agents, respectively. Fig. 8 depicts the mean savings rate in our experiment, where the mean savings rates of SCTP agent over TCP agent for packet loss rate, bandwidth, and round trip time are 11.9 %, 20. 8 %, and 5.5 %, respectively. To summarize, we can reduce the mean response time of TCP web agent by 12 % on average using SCTP web agent.



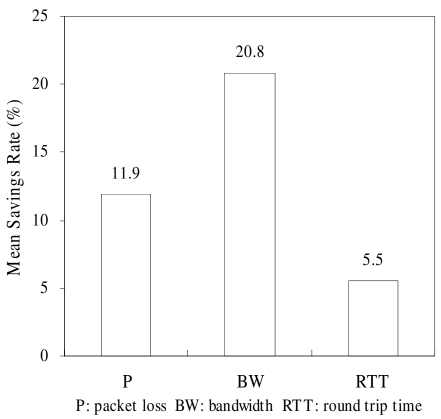


Fig. 7. Mean response time as a function of RTT **Fig. 8.** Mean savings rate of SCTP over TCP

4 Conclusions

In this paper, we have proposed a Stream Control Transport Protocol-based web agent. The web agent supports transport layer mobility and can overcome many of the limitations of network layer-based schemes. To demonstrate the efficiency of our web agent, we have carried out experiments in our experimental testbed. Simulation results show that our SCTP-based web agent can reduce the mean response time over a typical TCP-based web agent by 12 %, on average. Future work includes performance evaluation of multi-homing effect in a wireless web environment.

References

1. Bai, H, Fu, S. and Atiquzzaman, M.: Transport Layer Design in Mobile Wireless Network. Design and Analysis of Wireless Networks. Nova Science Publishers (2004).
2. Jamalipour, A.: The Wireless Mobile Internet. John Wiely & Sons Ltd. (2003) 368-384.
3. Caro, A., Iyengar, J., Amer, P., Ladha, S., Heinz, G. and Shah, K.: SCTP: A Proposed Standard for Robust Internet Data Transport. IEEE Computer. (2003) 20-27.
4. Fu, S. and Atiquzzaman, M.: SCTP: State of the Art in Research, Products, and Challenges. IEEE Communication Magazine. (2004) 64-76.
5. Alamgir, M, Atiquzzaman, M., and Ivancic, W.: Effect of Congestion Control on the Performance of TCP and SCTP over Satellite Network. NASA Earth Science Technology. (2002).
6. Al, A., Saadawi, T. and Lee, M.: Improving Throughput and Reliability in Mobile Wireless Networks via Transport Layer Bandwidth Aggregation. Computer Networks, Vol. 46. (2004) 635-649.
7. Stewart, R. et al.: Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. IETF Internet draft, draft-ietf-tsvwg-addip-sctp-06.txt. (2003).
8. Koh, S., Chang, M. and Lee, M.: mSCTP for Soft Handover in Transport Layer. IEEE Communication Letters, Vol. 8. (2004) 189-191.
9. Fu, S., Ma, L., Atiquzzaman, M., and Lee, Yong-Jin: Architecture and Performance of SIGMA: A Seamless Mobility Architecture for Data Networks, IEEE International Conference on Communi- cations (ICC), Seoul, Korea, May 16-20. (2005).
10. Unurkhaan, E., Pathgeb, E. and Jungmaier, A.: Secure SCTP- A Versatile Secure Transport Protocol. Telecommunication Systems, Vol. 27. (2004) 273-296.
11. Linux kernel stream control transmission protocol (lksctp): http://lksctp.sourceforge.net/
12. NIST NET Emulator: http://snad.ncsl.nist.gov/nistnet/.
13. Etherreal Protocol Analyzer: www.ethereal.com/.

A Network-Aware Truncating Module for Scalable Streams Saving Bandwidth for Overused Networks

B. Mathieu, Y. Carlinet, D. Massaloux, B. Kövesi, and D. Deleam

France Telecom R&D, 2 av, Pierre Marzin, 22307, Lannion, France
bertrand2.mathieu@francetelecom.com

Abstract. Wireless networks such as WiFi (802.11) enable new kinds of applications when users are in a mobility situation, including audio/video streaming. However, the total available bandwidth may vary greatly because of entwork usage and physical interferences in the 2.4GHz band. This may lead to a point where the bandwidth is overused causing a quality of service degradation. Based on the programmable node concepts and the scalable coding techniques, the functionality of a WiFi access point was extended to make it "*active*" and adaptive to the network conditions. Furthermore, since a WiFi access point is "*light*", the focus was to design this gateway as light as possible. Thanks to a module that monitors the wireless network and to a module that can truncate packets when the network is overused, the network-aware wireless gateway can adapt data according to the network bandwidth. This paper describes the designed scalable coding, the programmable wireless gateway and presents an evaluation of the gain of bandwidth after truncation of packets.

1 Introduction

Wireless networks such as WiFi (802.11) enable new kinds of applications when users are in a mobility situation (hotspot, hotel, airport….). Those applications include internet browsing, e-mails, file sharing, VoIP, audio/video streaming… However, the total available bandwidth may vary greatly because of network usage and physical interferences in the 2.4GHz band. This may lead to a point where the bandwidth is overused, resulting in increased packet loss and hence poor quality of service.

The idea is to design an element located at the WiFi access point, that can intercept in real-time all the packets of the same flows and apply an application-specific processing in order to reduce the bandwidth used when needed. Thanks to their very flexible and adaptive format, the scalable encoded streams [1] provides an attractive solution for audio/video streaming applications, for which the processing mentioned above consists in packet truncation.

Based on the programmable node concepts [2][3], we extended the functionality of a wireless gateway (a WiFi access point) to make it "active". This involves two modules: the first monitors the wireless network while the second may truncate packets (remove parts of the scalable stream). Furthermore, since the active gateway is based on the concept of programmable nodes, the "applicative" modules (e.g. the truncating module or even a transcoding module…) may be dynamically deployed and removed. For instance, in our case, when the network is not overused, the truncating module can be removed since it is not used.

This paper is structured as follows: After the introduction of the problem in Section 2, section 3 and 4 describe the scalable coding and the programmable gateway. The implementation is presented in section 5. Section 6 introduces the potential saving of bandwidth using this architecture while section 7 compares our solution to existing solutions. Finally, section 8 concludes this paper and presents the future work.

2 Problem Statement

Wireless networks such as WiFi Hot-Spots are built around small and light equipments, called access points, which enable users of the wireless network to communicate between each other and also with users located outside the wireless network (Internet for instance). An access point has then two interfaces: one wireless interface (802.11) and one wired (fixed) interface (fig. 1).

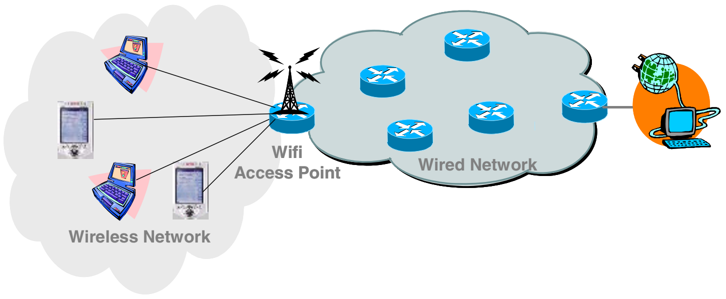


Fig. 1. WiFi Network Architecture

The bandwidth is shared between all users, and thus it may vary depending on the number of connected users and on the type of used applications; downloading is more bandwidth consuming than browsing…Then, if the network is overused, lot of packets may be lost and thus the quality of service perceived by the end-users is very bad.

This paper focuses on audio/video streams. The aim was then to find a solution that, in the case of overused network, can reduce the used bandwidth of streaming applications with little degradation of the quality perceived by the end-users.

Obviously, a way to gain bandwidth is to transmit less data on the wireless networks when the latter is overused. It can be performed by dropping packets. It may be done based on policies like Gold users, Silver users and depending on the user subscription to the service. Yet it is not acceptable for the services where users must share the same quality of service. Another solution is to drop packets randomly. But in this case, the quality of service for given users may be very bad if lot of the packets sent to them are dropped.

Finally, very well-suited to streaming applications, the use of scalable streams appears to be the best solution to reduce the used bandwidth with low quality degradation. The truncation of refinements layers in data packets enables to reduce the size of packets with a decrease of the quality depending on the amount of discarded bits. This functionality has been implemented in a network element located between the server and the end-users, namely the access point.

In current hotspot infrastructure, access points can provide network facilities (DHCP server, NAT, even firewall) for end-users devices located in this wireless network but they do not process data at the application level. Extending the functionalities of such access points by integrating some concepts emerging from the programmable network community was then the chosen solution. This means that the "new" programmable gateway (the term *gateway* is now preferred in order to avoid confusion with current access points) must be able to process data transparently at the application level (without splitting the client-server connection), and must provide an environment enabling the dynamic deployment of modules.

Prior to the description of the programmable gateway, the next section describes the implemented scalable codec.

3 Scalable Coding

In scalable coding the coded information is organised in layers: a core layer contains the minimum information needed to obtain a basic quality decoded signal. Each successive layer contains new information on the same signal part and improves the previously obtained quality.

Scalable encoders process and send the full range of layers, but the decoders do not necessarily receive all the layers. The decoders process the received layers with an obtained quality roughly proportional to the number of decoded layers.

Scalable codecs have been proposed for a long time in the state of the art: an embedded scheme like ITU-T G.727 (16/24/32/40 kbit/s) is an example of scalable coder. For such coders, all the layers produce signals of the same bandwidth, and the scalability is obtained at the price of degradation in terms of performance when compared to fixed rate schemes. Other scalable codecs increase the bandwidth with added layers. Quite commonly also, the core layer coding scheme is different from that of the other layers: the core codec techniques provide their best performance at low bitrates, whereas extension layers use methods more suitable for higher bitrates. This allows covering a large range of bitrates and qualities, with sufficient quality at relatively low rates and high quality at higher rates. The MPEG4 CELP-AAC presented in [4] is a good example of a scalable structure with two different techniques in the core and extension layers.

In all those existing scalable codecs, the bitrates allocated to the layers are defined in advance, and the granularity depends on these bitrates. Test results also show that the performance of such scalable codecs is generally lower than those using the same coding technique optimised for the same fixed bitrate.

The scalable codec presented here operates on mono signals in the 12.2 kbit/s to 31,4 kbit/s bitrate range. The maximum codec bitrate (31,4 kbit/s) has been arbitrarily chosen to get a 32 kbit/s bitrate when including the RTP payload header that is commonly used. This maximum bitrate value could be higher (or lower) since the codec structure is very flexible and not hard defined. At 12.2 kbit/s the bitstream produced by the scalable codec is compatible with the ETSI/3GPP standard AMR [5]. At that bitrate, this bitstream is also compatible with the current GSM network codec (GSM-EFR), providing good quality over speech signals in narrowband ([300, 3400 Hz]). When the bitrate increases, the bandwidth is enlarged to wide band ([50, 7000 Hz]) and the quality increases (both for the low and high bands). For instance, performances on music signals are much better at higher bitrates.

The scalable codec is a combination of two cascaded encoding schemes: the first one (core coder) is the ETSI/3GPP AMR CELP coder. The second one is a transform (MDCT) coder, developed within France Telecom laboratories under the name of 'TDAC coder' ([6]).

Notice that since the AMR is a multi-rate codec, one of the other seven bitrates could be selected for the core codec instead of 12.2 kbit/s.

The first part of the bitstream containing the bits of the AMR codec complies with the RTP payload format (byte aligned, including Table Of Content TOC). This ensures the compatibility with a pure AMR terminal.

The frame length of the scalable codec is 60 ms, corresponding to three AMR and TDAC coder frames. The total 60 ms frame size in bytes is 240.

The encoder has a split band structure. The wideband input signal sampled at 16 kHz is first down sampled to the narrowband. Each narrowband 20 ms block of input signal is encoded by the AMR codec. Next, the AMR decoded signal is up sampled to wideband, and the difference between this signal and the original one is encoded using the TDAC coder. The corresponding decoding scheme is as follows: the three AMR frames are first decoded, then the three TDAC frames are decoded using the decoded AMR samples and producing the final output.

The scalable codec uses an efficient frame erasure concealment algorithm ([7]) that can correct the effect of missing frames even in severe packet loss conditions.

A more detailed description of the codec can be found in [8]. One of the main advantages of this scalable scheme is its high granularity: the bitstream can be truncated at any desired length, in practice, at the byte level, with a quality statistically corresponding to the number of received bits. Therefore, the algorithm is quite flexible in terms of bitrate scalability. It also limits performance loss due to scalability: informal test results have shown that at lower bitrates, the presented scalable coding scheme performs equally or even better than the equivalent non scalable version where the bit allocation is made for the given bitrate.

4 Programmable Gateway Architecture

The proposed architecture is inspired by programmable network principles [2][3] that form a prerequisite for deploying third-party code in the network of a network operator under the control of the latter.

Applying these principles to the gateway design, 3 layers have been defined. The lower layer monitors the wireless bandwidth and routes/redirects incoming packets. The second layer manages the applications and lower layer components. It checks if data processing is needed (when network is overloaded) and configures the lower layer to redirect packets if needed. The third layer modifies the data themselves to fit the network requirements (truncation at the best layer). Furthermore, in this layer, the proposed architecture encompasses dynamic deployment mechanisms in order to be able to deal with different formats if the truncation is no longer efficient (transcoding with another codec, allowing a lower bandwidth use, may be deployed). The figure 4 presents this architecture.

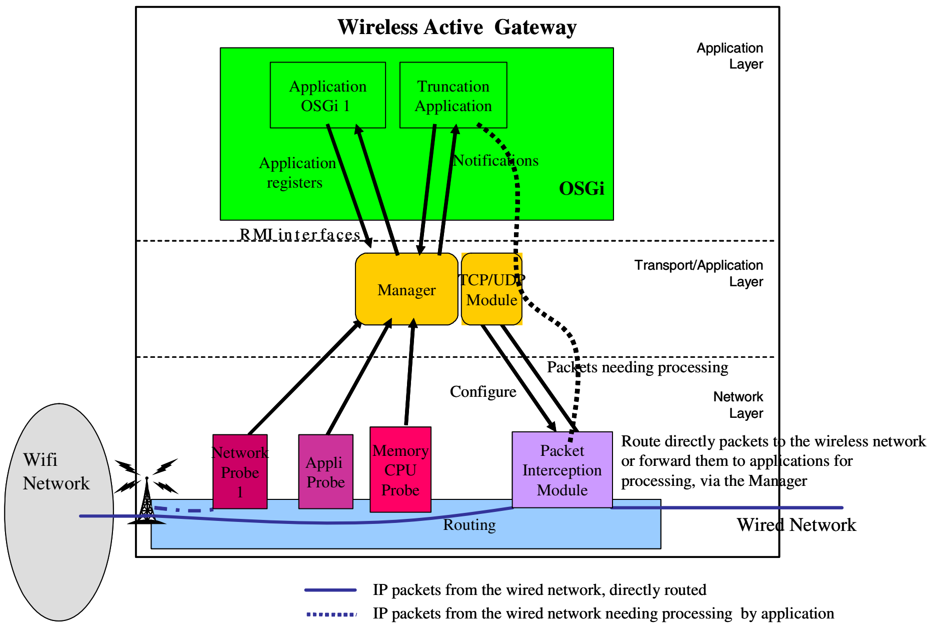

Wireless Active Gateway
Application Layer
Application OSGi 1
Truncation Application
Application registers
Notifications
OSGi
RMI interfaces
Transport/Application Layer
Manager
TCP/UDP Module
Packets needing processing
Configure
Network Layer
Wifi Network
Network Probe 1
Appli Probe
Memory CPU Probe
Packet Interception Module
Route directly packets to the wireless network or forward them to applications for processing, via the Manager
Routing
Wired Network
IP packets from the wired network, directly routed
IP packets from the wired network needing processing by application

Fig. 4. Programmable Gateway

Each component of the gateway is now described in more detail.

- **Network Probe**

Since the gateway must process data (truncate packets) only when the network bandwidth is overused, a component that can monitor this bandwidth is needed. All the packets pass through the gateway (infrastructure mode) and there is a module that counts the number of sent/received bytes on the wireless network. The network probe gives feedback on the used bandwidth to the component "Manager".

- **Packet Interception Module**

Packets must be truncated only when the network is overused. It is then required to have a component that can perform the redirection of packets to the application in the user space or the forwarding of the packets to the wireless network. Furthermore, this component must be dynamically programmable/configurable since the redirection depends on the network load which is very variable. In the gateway architecture, the "Manager" is the only element that configures this component, upon request from the application.

- **Manager**

The Manager is the central component of the gateway, managing tables of applications with their defined thresholds and eventually configures rules in the Packet Interception Module. The application registers to the Manager to be notified when the used bandwidth exceeds one or several thresholds. The manager knows the used bandwidth via its close interaction with the network probe and then can notify the application if the current value reaches a threshold defined by the application.

When receiving the notification, the application may take several decisions like requesting to receive packets and perform processing on the packets. It is the case for the truncating module that will truncate packets more or less severely (a lot of or few bytes in the packets) depending on the used bandwidth.

The manager is then responsible for configuring the "Packet Interception Module" to redirect incoming packets to the application.

- **« Truncating » Module & Execution Environment**

This is a module that applies application-specific processing on the packets. The packets are received and then forwarded to the Packet Interception Module. To ensure the programmability of the gateway, it was decided to use an execution environment that provides dynamic deployment facilities. The truncating module may then be deployed only when needed.

5 Implementation Issues

This section focuses on the implemented modules of the gateway, mainly specifying the used technologies and their internal relationships.

The implementation was driven by the constraint of the lightest possible design for the gateway, so that truncation is performed on a light equipment. The Linux Operating System has been chosen, configured to act as a router in order to route packets from the wired network to the wireless network, an OSGi [9] framework to deploy applicative modules and a light java virtual machine CDC (Connected Device Configuration) HotSpot (formerly CVM : C Virtual Machine) [10].

- **Network Probe**

The network probe is developed based on the "pcap" library [11]. This library enables to capture packets passing through the wireless adapter of the gateway. The network probe adds the time to the packets and is able to estimate the used bandwidth per seconds. This value is periodically fetched by the Manager.

- **Packet Interception Module**

The implementation is based on the Linux "Netfilter" mechanism. Netfilter [12] is a component, efficient for intercepting packets and passing them to a user-level application via ip_queue, or forwarding them directly to the wireless network. The behaviour of Netfilter is defined by rules, based on IP addresses, IP ports, protocols... The "Manager" component is in charge of configuring Netfilter. In case of normal use of the wireless network (network not overused), no rule is applied and then packets are directly forwarded, performed by this component running in the Linux kernel, leading to a high performing routing task. If rules are defined, packets concerned by these rules are forwarded to the Manager that forwards them to the right application.

- **Manager**

The Manager is implemented in Java, communicating with the application through the use of RMI (Remote Method Invocation). In order to have a light equipment with good performances in spite of low CPU, it was decided to employ the CVM virtual machine, lighter than a traditional J2SE Java Virtual Machine. The Manager is a Java program running inside the CVM.

- **« Truncating » Module & Execution Environment**

The applications are developed in Java and designed to be running in an OSGI framework. The OSGi (Open Services Gateway Initiative) framework, designed at the OSGi Forum, has been chosen as execution environment for its lightness (OSGi is designed for light equipments like car, home gateway…). Indeed, since the objective is to use such a gateway in light equipments like access points, a small footprint is necessary. Furthermore, one interesting feature is its ability to dynamically deploy new modules and to resolve dependencies between bundles. This feature enables us to deploy the truncating module only when needed (overused network) and for further issues, the idea is to deploy others "saving bandwidth" modules, not just only for media streams but also for others types of traffic (like adaptation for HTTP traffic for instance…), depending on the type and amount of traffic on the wireless network. The "Jeffree" [13] OSGI framework was chosen for implementing the execution environment of the programmable gateway.

Since the used scalable codec produces one-byte layers, the truncation may be performed at every level (a fine-grained truncation). In our case, 3 levels of truncation and thus 3 associated thresholds have been defined and depending on the value of the used bandwidth, the truncation is performed at different locations inside the packet. The truncating module communicates with the Manager through RMI interfaces.

6 Evaluation and Future Work

This section presents the amount of bandwidth that is reduced using truncation in the programmable gateway whereas the quality is moderately decreased.

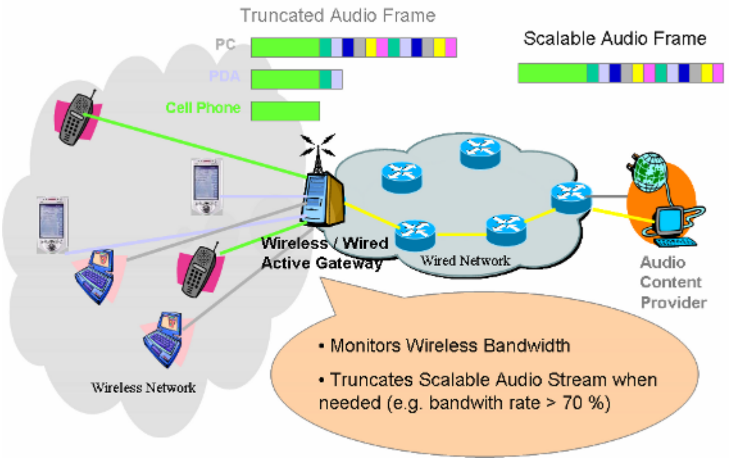

Truncated Audio Frame
PC
PDA
Cell Phone
Scalable Audio Frame
Wireless / Wired Active Gateway
Wired Network
Audio Content Provider
Wireless Network
• Monitors Wireless Bandwidth
• Truncates Scalable Audio Stream when needed (e.g. bandwith rate > 70 %)

Fig. 5. Evaluation scenario

We have setup a testbed including an audio streaming server (simulating a Web radio for instance) on the wired network, a wireless programmable gateway interconnecting the wired network to the wireless networks. Finally, one computer simulates end-users receiving the streams (listening to the Web Radio).In the implementation, the audio stream is encoded using the scalable coding described in section 3, with a core fixed at 12,8 kbit/s (96 bytes : core codec at 12,2 kbit/s plus RTP payload header) and refinements layers up to 32 kbit/s (240 bytes).

With the increase of end-users receiving this stream, the bandwidth use will increase linearly and then lead to the overuse of the wireless network. Thereof, several actions can be taken, like truncations at different levels. The following figure presents the gain of bandwidth when truncating packets at different levels.

Table 1. Used Bandwidth When Truncating

Nb users	Initial : 240 bytes	Truncation: 180 bytes	Truncation: 135 bytes	Truncation: 96 bytes
1	32	24	18	12.8
10	320	240	180	128
20	640	480	360	256
40	1280	960	720	512
60	1920	1440	1080	768
80	2560	1920	1440	1024
100	3200	2400	1800	1280
120	3840	2880	2160	1536
140	4480	3360	2520	1792
160	5120	3840	2880	2048

The numbers presented here only refer to the bitrate of the audio stream, not including the packet header (like IP/UDP/RTP Header).

Those values only involve the audio stream but in the wireless network, users may use other applications (file-sharing, video streaming..) and thus generate extra traffic (compared to the given table) leading more rapidly to an overused network. The table 1 just aims at showing the gain of bandwidth and not the maximum number of end-users needed to saturate the wireless network.

Furthermore, these values are theoretical and need to be confirmed by real tests. We have performed tests simulating the end-users on only one computer but this evaluation is limited since only one wireless network adapter is used and then the access to the media does not really simulate the behaviour of the wireless network when a lot of users are connected to it. This mainly relates to the time for the wireless network adapter to check if it can send data (no other adapters sending), to wait for collisions…To summarize, all features designed in the CSMA/CD (Carrier Sense Multiple Access / Collision Detection) [14] standard, like the DIFS (DCF InterFrame Space), the BackOff Interval, need to be really taken into account when doing further simulations to have a correct view of the bandwidth gain.

7 Related Work

The used bandwidth reduction may be achieved using multicast streams. However in our case, we can perform adapation of the streams on a per-terminal basis, which is not possible with mutlicast.

A lot of studies try to find a solution for a better Quality of Service on wireless networks. They mainly focus on the improvement of the transport protocol (using TFRC mechanism or improvements for instance) but they do not take into account applicative data [15][16] ….

Some papers cope with this issue at the application level, by using transcoders but they mostly use it when multimedia codecs between end-devices are incompatible [17] or by changing the bit-rate to adapt to the bandwidth usage [18][19]. However, those processing's are complex (in terms of CPU), increase delay, degrade quality and they can not be implemented on a "light" equipment like an access point.

A current solution that copes with applicative data, with lesser CPU-consumption is the scalable coding [1] managed by the end-users but with no intermediate element, located between the end-users and the content server. The more the data are decoded by the receiver, the better the quality will be. This architecture allows for decoding of only part of a total stream but does not prevent the bandwidth overuse since the total packets are emitted over the transmission network. Furthermore, the sending rate may not be reduced on the sender's side since some users may be able to receive all the data (those who are not located in an overused network) in the case of a multicast transmission.

8 Conclusion

In this paper, a novel network architecture, including a programmable wireless gateway to adapt data to the network conditions, has been presented. The gateway includes a component that monitors the wireless network in real-time to evaluate the used bandwidth. A manager compares this value to thresholds given by applications and if it is above, configures a Packet Interception Module to redirect incoming packets to the application. A major innovation of the proposed architecture is the use of scalable encoded streams that are truncated by the applicative modules within the gateway when the network is overused. The paper shows the benefits of scalable streams (that allow significant bitrate decrease with slight degradation of the quality, as perceived by people used to listen to audio streams for measuring the audio quality) and the advantage of doing truncation in a network-aware element when the network bandwidth varies greatly. The future work is to perform real tests to verify the benefit of such an approach and to extend this concept of others types of traffic, for instance to save bandwidth also for HTTP traffic by the use of adaptation modules (like XML/XSL transformers) or for SMTP, sending just the title and the first lines of e-mails when the network is overused… This could be designed to share the "reduced" quality amongst several applications and not just media streaming.

References

1. 1. H.Taddei, D.Massaloux, A.Le Guyader. "A Scalable Three Bitrate (8, 14.2 and 24 kbit/s) Audio Coder". *The 107th Convention AES*, New York, September 1999.
2. Andrew T. Campbell, Herman G. De Meer, Michael E. Kounavis, Kazuho Miki, John B. Vicente, and Daniel Villela, "A Survey of Programmable Networks", *ACM Computer Communications Review,* April 1999
3. A. Lazar, "Programming Telecommunication Networks", *IEEE Network*, September/October 1997, pp. 8-18

4. Grill. "Implementation of a scalable audio coder based on AAC Modules and a core model". *ISO/IEC JTC1/SC29/WG11 MPEG97/1816*. February 97.
5. B.Bessette, R.Salami, R.Lefevre, M.Jelinek, JR.Pukkila, J.Vaino, H.Mikkola, K.Järvinen."The Adaptive Multi-rate Wideband Speech Codec (AMR-WB)". *IEEE Trans. On Speech and Audio Processing,* Vol.10 n°8, November 2002.
6. Y. Mahieux, J.P. Petit, " High quality audio transform coding At 64 kbit/s," *IEEE Trans. on Com.*, Vol.42-11, Nov.1994, PP.3010-3019.
7. B. Kövesi, D. Massaloux : "Method of packet errors cancellation suitable for any speech and sound compression scheme" *ETSI STQ- Workshop on packet loss compensation* – Nice, France, February 11, 2003
8. B. Kövesi, D. Massaloux and A. Sollaud, "A scalable speech and audio coding scheme with continuous bitrate flexibility", *ICASSP2004*, Montréal, May 2004
9. OSGI Forum: www.osgi.org
10. CDC HotSport : (formerly CVM): http://java.sun.com/products/cldc/
11. pcap : Tim Carstens, Programming with pcap; http://www.tcpdump.org/pcap.htm
12. Linux Netfilter : http://www.netfilter.org/
13. Jeffree : http://jeffree.objectweb.org/
14. 802.3 CSMA/CD standard : http://standards.ieee.org/getieee802/802.3.html
15. Minghua Chen, Avideh Zakhor, " Transmission Protocols for Streaming Video over Wireless ", *IEEE ICIP 2004*, Singapore, October 24-27, 2004
16. Jörg Widmer, Mark Handley, " Extending Equation-based Congestion Control to Multicast Applications", *Sigcomm 2001*, San Diego, USA, August 27-31
17. B.Mathieu and al, “Deployment of Services into Active Networks”, *WTC-ISS 2002*, Paris, France, 22-27 September 2002
18. Z.Lei and N.D.Georganas, "Rate Adaptation Transcoding For Video Streaming Over Wireless Channels", *In Proceedings of IEEE ICME 2003*, Baltimore, USA, July 6-9
19. Abdelbasset Trad, Qiang Ni, and H. Afifi, " Adaptive VoIP Transmission over Heterogeneous Wired/Wireless Networks", *MIPS 2004*, Grenoble, France, November 17-19, 2004

APHIDS++: Evolution of A Programmable Hybrid Intrusion Detection System

Mohammed S. Alam, Abhishek Gupta, Jake Wires, and Son Vuong

Department of Computer Science,
University of British Columbia,
Vancouver, Canada
{malam, agupta, jtwires, vuong}@cs.ubc.ca

Abstract. With the rapid growth of the Internet and the ever-increasing security problems associated with its popularity, the need for protection against unwanted intruders has become imperative. Antivirus software, intrusion detection systems, spyware detectors, and mal-ware detectors are some of the protection mechanisms available to users today. The diversity of these manifold systems suggests the need for a unifying managerial system, such as APHIDS (A Programmable Hybrid Intrusion Detection System), which can correlate and coalesce preexisting security components. In this paper we provide a description of improvements made to the initial APHIDS design, comprising the introduction of agent caching, the addition of an optional intelligent agent, and an XML implementation of our Distributed Correlation Script (DCS).

1 Introduction

Intrusion detection systems (IDS) are rapidly being deployed in both industry and academia to provide the best possible protection of personal and confidential information. Because different intrusion detection systems are good at detecting different kinds of attacks, it makes sense to combine the strengths of various existing IDSs to form a general, overarching security solution. This is the main philosophy underlying the original APHIDS system [1].

This paper reports our contributions in extending the original APHIDS architecture. Notable extensions include the incorporation of agent caching, the addition of an optional intelligent agent, and the modification of our original proposed Distributed Correlation Script (DCS).

The remainder of this paper is structured as follows: Section 2 provides background information about related work in intrusion detection using mobile agents, along with a brief introduction to the original APHIDS system. Section 3 describes the implementation of agent caching in APHIDS++, and Section 4 describes the addition of our intelligent policy agent. Section 5 provides the rationale for our inclination towards using an XML based scripting language for DCS. Finally, we conclude the paper by presenting details about our intended future developments to the APHIDS system.

2 Related Work

Intrusion detection systems are usually categorized into two groups: anomaly and misuse detectors. In anomaly detection, a system is first trained to detect normal traffic patterns. Once the system has been inculcated with a recognition of normal traffic, it can raise alarms upon the observation of anomalous behaviors. On the other hand, misuse detectors contain preestablished misuse signatures, which, when matched, detect an intrusion event. One such freely available misuse detection system is SNORT [2].

Previous mobile agent based intrusion systems include the AAFID project [3], the IDA system [4], the Micael system [5], the Sparta system [6] and the MA-IDS system [7]. The mobile agent based approach used in APHIDS shares a few similarities with the Sparta and MA-IDS system [1].

2.1 APHIDS Background

Perhaps the most salient aspect of APHIDS, and the mechanism by which it can out-perform conventional IDS correlation techniques, is its use of mobile agents [1]. The correlation of IDSs often involves mining large repositories of data stored locally at IDS machines. Mobile agents allow APHIDS to perform such investigations on-site, thereby obviating the need to transmit large quantities of data across the network. A single programmed agent can thus evaluate large data logs at a number of different IDSs simply by migrating from machine to machine as required. This very idea is the key motivation behind the APHIDS architecture. In effect, the APHIDS system, implemented in Java atop the mobile agent platform Grasshopper [12], is a mobile agent based intrusion detection framework which integrates existing IDS technologies to retrieve more detailed information about an intrusion [8].

The various agent types defined in [8] have been included in Table 1 for reference. Figure 1 shows a high level relationship between the agent types. The subsequent sections discuss the new additions made to this initial APHIDS system.

Table 1. This table gives a brief definition of the functionality of the various agent types mentioned in [8]

Agent type	Functionality
UI	Provides the user interface for the entire system.
System	Accepts inputs from the UI agent to control correlation agents.
Correlation	Receives trigger events from the trigger agent. Launches task and action agents. Is responsible for correlating data collected.
Trigger	Sends trigger events to correlation agent on intrusion detection.
Task	Launched by the correlation agent to gather data from remote hosts.
Action	Launched by correlation agent to perform required final actions.

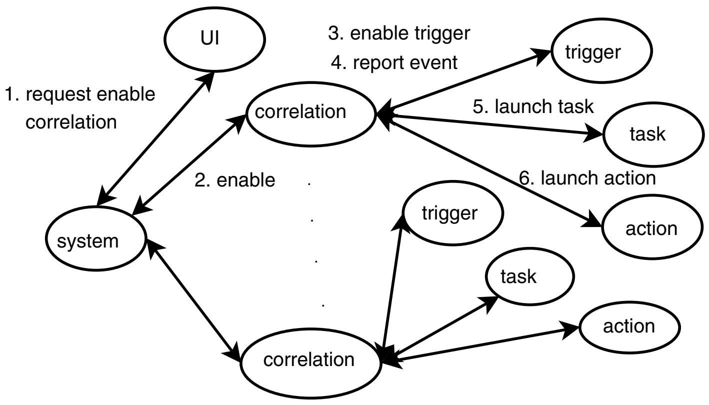


Fig. 1. This diagram shows the interaction between the various agents in the initial APHIDS implementation [8]

3 Agent Caching

Given that the primary advantage of an agent-based system is the code mobility it affords, the concept of agent caching might initially appear to be counterintuitive: the practice of caching involves maintaining quiescent agents locally to render later agent migrations unnecessary. However, caching does not annul mobility within the APHIDS framework, nor does it reduce the advantages obtained therefrom. Rather, it simply eliminates redundant efforts within the system. As already mentioned, the real benefit of agent-based systems arises from the ability to move code, which is typically relatively small, to data, which is often quite large. Even with caching enabled, APHIDS capitalizes on this benefit: agents still migrate to perform computations locally, and the savings gained by such mobility is in no way diminished through caching.

While caching does nothing to detract from the mobility of APHIDS, it makes considerable contributions to the system's performance. The principal costs involved in reacting to an attack within the APHIDS framework are twofold: the first arises from the time required to instantiate a new agent, and the second derives from the time and bandwidth required to deploy the agent. On the occurrence of a cache hit in a cache-enabled system, the first cost can be eliminated entirely (because an already extant agent can be reused) and the second reduced dramatically (because the amount of code required to update an agent is considerably less than the agent code itself).

The benefits of agent caching, as is the case with all types of caching, are only to be enjoyed if a certain locality exists: the efficacy of agent caching is entirely dependent on a pattern of repeated attacks. It is likely, however, that having gone to the trouble of attacking a machine once, a malicious assailant may be inclined to do so repeatedly. Such behavior, in fact, is fundamental to the Denial of Service (DoS) attacks whose prevalence within the Internet distresses network administrators. In the original APHIDS system, a sustained assault on a single machine would require the instantiation and transmission of a separate (but identical) agent for each individual attack. With caching in place, the amount of work performed by APHIDS in the case of such an assault is drastically reduced: a single agent can be sent to the afflicted machine to handle each attack sequentially. Moreover, the opportunity for load balancing and dynamic tuning exists, as the Correlation Agent responsible for the creation and deployment of agents can vary the number of attacks to be handled by each cached agent: as this parameter is decreased, more agents are deployed (consuming more processor and network resources), but more attacks can be handled in parallel (reducing response latency and increasing throughput), and vice versa as the parameter is increased.

3.1 Cache Design

The managerial agent in the APHIDS system is the Correlation Agent. It is this agent's responsibility to instantiate and deploy Task Agents upon the reception of Trigger Agent alerts, which notify the Correlation Agent of current attacks. Task Agents then migrate to the pertinent machine to perform the necessary computations. In the absence of caching, these agents would terminate after reporting the results of their calculations back to the Correlation Agent. When caching is enabled, however, these Task Agents remain at their destination machines to serve subsequent attacks and their locations are maintained by the Correlation Agent in a data structure known as the APHIDS Cache.

APHIDS++ employs a two-level caching scheme. After having handled an initial attack, Task Agents will enter the first level cache mode, which consists of a busy wait at the attacked machine. Each Task Agent maintains a publicly accessible queue of pending attacks to handle; if the Correlation Agent receives a trigger corresponding to another attack at the same location as the cached agent,

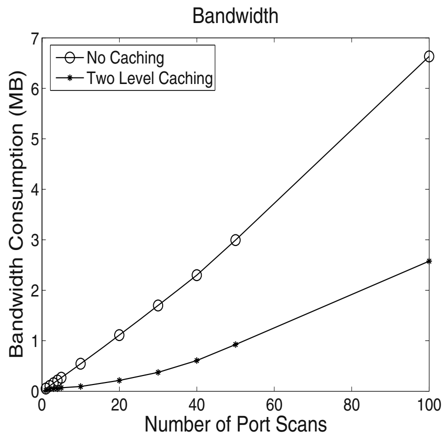


Fig. 2. Bandwidth Comparison

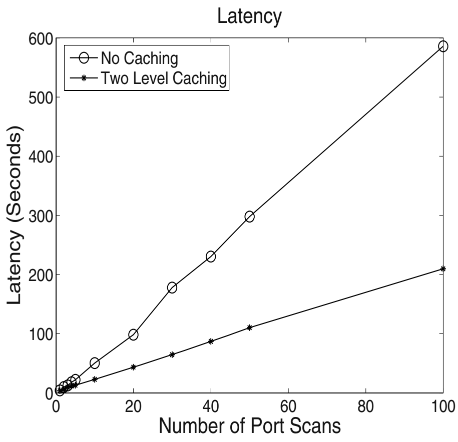


Fig. 3. Latency Comparison

a new agent is not deployed; rather, the information describing the important aspects of this attack is transmitted across the network and inserted into the cached agent's queue.

While in the first cache level mode, the Task Agent will perform a busy wait on this queue until it is provided with another attack to handle. Having received the information describing a new attack, the Task Agent will again perform the relevant computations and report the results back to the Correlation Agent. If no new attacks are sent to the Task Agent within a certain time limit, known as the *SpinLifeTime*, the agent proceeds to the second cache level mode, in which it is flushed to its host machine's disk. At this point, the agent is no longer consuming the host machine's memory or processor time, but is still able to receive new attack descriptions.

If a new description is sent to a flushed agent, the agent is reloaded into memory and handles the attack accordingly, and then reenters the first level cache mode (i.e. it performs a busy wait, with a reset *SpinLifeTime*). If no new attacks are sent to a flushed agent within a time interval known as its *FlushedLifeTime*, the agent is terminated from its host and removed from the APHIDS Cache.

3.2 Evaluation

We tested our system against a flurry of port scan attacks and observed its behavior as compared to the original APHIDS implementation. For comparative purposes we use bandwidth consumption and system response time (latency) as parameters of interest. Bandwidth consumption was measured by capturing all pertinent packets related to Aphids operation-all communication between port 7000 on the APHIDS host machine and the machine hosting the Task Agents was measured. To avoid clock synchronization issues, all latency measurements were performed on the APHIDS console machine. In particular, the time elapsed between the reception of the trigger and the response from its corresponding Task Agent formed the basis of our latency measurements.

In Figure 2, we plot the bandwidth consumed by the system against the number of attacks. Due to the implementation of the SNORT IDS used, each single port scan shown was actually registered as three simultaneous attacks. It can be clearly observed that both systems are scalable and maintain a linear relation between bandwidth consumption and the number of attacks. However, by using caching we can reduce bandwidth consumption to approximately one third of what the original system consumes.

In Figure 3, we plot the response time or latency of the system against the number of attacks. Again, each port scan here actually represents three simultaneous attacks on the system. Both systems responded linearly to the flurry of attacks; however, by using our two-level caching scheme we observed an improvement of 300% in the response time.

The significant improvements in bandwidth and response time were expected. In the original APHIDS design every time an attack is detected a new Task Agent is launched by the Correlation Agent for mining data at the IDS Host; in APHIDS++, Correlation Agents need merely send descriptive messages to cached agents upon the occurrence of repeated attacks. Since the size of a message is considerably smaller than the size of an agent, we enjoy a substantial reduction in bandwidth consumption. The reuse of existing agents in the network also leads to improvements in system response time, because the time necessary to instantiate and migrate a new Task Agent is much larger than the time required to send a simple update message.

4 Intelligent Policy Agent

The Policy Agent is a core component of APHIDS++. It was introduced to coordinate the behavior of Trigger and Correlation Agents, allowing the system to avoid performing superfluous correlation tasks, a profusion of which may ultimately lead to denial of service attack vulnerabilities. The Policy Agent operates by consulting a set of rules, which it modifies continually, thereby learning from previous intrusion attempts.

Our motivation for incorporating a Policy Agent in APHIDS++ follows directly from the Blaze system [9]. Blaze is a theoretical IDS designed specifically for VoIP (Voice over Internet Protocol). Contrarily, APHIDS++ is a generic IDS suitable for a variety of applications, including VoIP intrusion detection. Figure 4 diagrams the interaction between the Policy Agent and the original APHIDS agents, while an approximate sequence of interactions between various agents in APHIDS++ is modeled in the following steps.

1. Trigger Agent detects a probable intrusion event.
2. The Trigger Agent sends a message regarding the type of intrusion event to the Policy Agent.
3. Upon receiving a message from the Trigger Agent, the Policy Agent consults its set of rules in order to decide whether the intrusion event should be reported to the Correlation Agent. The decision is then sent, along with additional data, as a notification to the Trigger Agent.
4. If the notification from the Policy Agent contains a request to forward the intrusion event to the Correlation Agent, the Trigger Agent prepares a message, containing the data collected in step 1 as well as the data sent to it by the Policy Agent, and sends it to the Correlation Agent.
5. The Correlation Agent then spawns one or more Task Agents to gather more data regarding the reported trigger event. The data returned by Task Agents is then correlated by the Correlation Agent.
6. The Correlation Agent then queries the Policy Agent for any information pertaining to the correlated data.
7. The Policy Agent sends the requested data to the Correlation Agent.
8. The Correlation Agent, after comparing the data collected from the correlated task and the data received from the Policy Agent, may send an update request to the Policy Agent to update its rules or data. These rules are best used by the Policy Agent while responding to similar future attacks in step 3. This, in effect, facilitates the Trigger Agent's intelligent handling of identical intrusion events.
9. The Correlation Agent finally invokes an appropriate Action Agent.

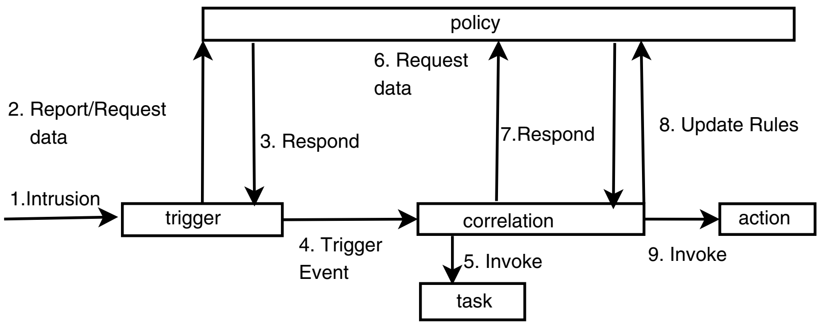


Fig. 4. Interaction between agents with the presence of Policy Agent in APHIDS++

The following section provides a description of how APHIDS++ can be used to detect intrusions in a VoIP attack scenario.

4.1 Scenario of Use in a VoIP Reinvite Message Attack

We have successfully used the model presented in the previous section to detect a simulated VoIP Megaco/H.248 reinvite message intrusion[9]. Initially the system is configured with the Trigger Agent placed at a media gateway (MG), as shown in Figure 5. The Correlation Agent and the Policy Agent are setup on

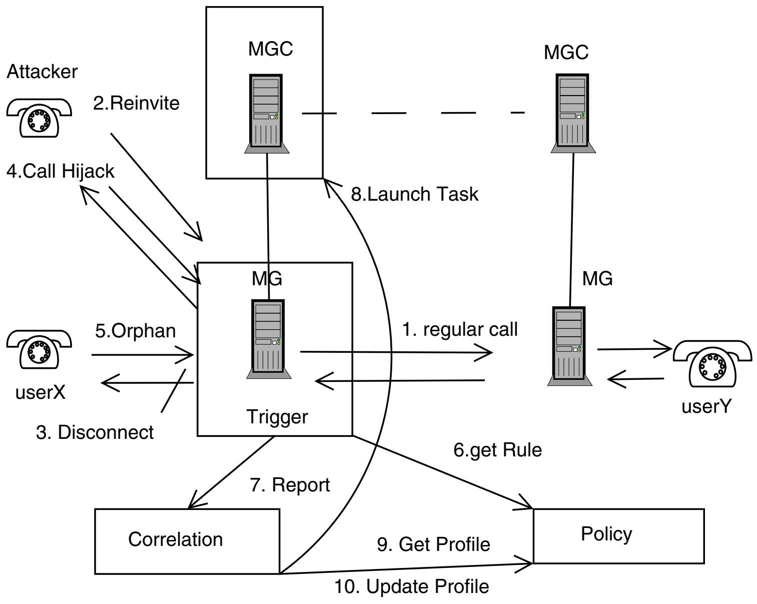


Fig. 5. Use of APHIDS++ in a VoIP reinvite attack scenario

two separate hosts. It is possible for an attacker to hijack the current call between user-x and user-y by using the user-id of user-x to send a reinvite message to user-y through the media gateway (MG). User-x now hears only static, and will not be notified of the call hijack in the current protocol. Moreover, he will not be able to make any subsequent calls unless he completely resets his VoIP system. If the user does not reset his system, he will continue sending *orphaned* packets (packets that will not be forwarded by the MG) meant for user-y. This acts as a probable intrusion event in our system, as mentioned in step 1 in the previous section.

Once the Trigger Agent, sitting at the MG, receives an orphaned packet, it sends a request to the Policy Agent enquiring if the source IP of the orphaned packet (user-x) is authorized to connect to the current MG. If the Policy Agent gives a negative feedback, then the Trigger Agent does not report a trigger event to the Correlation Agent. This has the advantage of blocking distributed denial of service attacks from hosts not authorized to connect to an MG.

On the other hand, if the Policy Agent determines that the source is valid, it sends a list of IP addresses maintained in user-x's profile [1]. This list includes all known IP addresses which have been previously used by user-x in making calls.

The Trigger Agent then inspects the current calls being routed through the MG, and tries to find a match to the IP list received from the Policy Agent. A matching IP address indicates that the potential attacker IP is actually a phone in use by user-x and that the source of orphaned packets is another instrument which user-x forgot to hang up. This mechanism for detecting a false alarm avoids the unnecessary invocation of Correlation Agent and subsequent Task and Action Agents.

On the other hand, if no matching IP address was found, the Trigger Agent reports a probable intrusion event to the Correlation Agent. The Correlation Agent, in turn, launches a Task Agent in order to query the MGC for more information regarding the reinvite message. Once the Correlation Agent receives a response from the Task Agent, it prepares a correlated user profile by correlating the information received from the Trigger Agent and the Task Agent. After correlation, it queries the Policy Agent to obtain the entire profile of user-x and then compares it with the new correlated user profile. A large hamming distance between the two profiles indicates an indubitable intrusion event, whereas a hamming distance below a user-configurable threshold denotes a false alarm. In the event of a false alarm, the Correlation Agent adds the spurious attacker IP to user-x's profile and sends it as an update to the Policy Agent. Subsequent uses of that IP are intelligently detected by the Trigger Agent as non-intrusive behavior.

[1] A user's profile is based on his past usage of the VoIP system and summarizes attributes such as the IP addresses of phones belonging to the user, usual times of day when the user makes a call, average duration of calls, number of calls made over a period of time, etc.

5 XML Based Distributed Correlation Script

In [1], the authors have mentioned the use of a Distributed Correlation Script (DCS), which is a high-level specification language used to describe an intrusion detection correlation routine. A DCS is required not only for describing and initializing the agents involved in a correlation routine, but also for defining the dynamic state of a correlation routine during runtime. The dynamic state is achieved by using implicit states and state-transition conditions.

In APHIDS++, we have used XML to implement our DCS. Using XML with mobile agents is not a new topic. It has been used in the XMARS system [10] for agent communication. We have used an open source library, Xerces (available at http://xml.apache.org/xerces-j/), for implementing our script parser.

There were many reasons for using XML in our system. First, the availability of a built in validation scheme in XML using DTD (Descriptive Type Definition) avoids complications that could develop due to incorrect DCS file input from a user. Second, the availability of XML visualization and construction tools such as XMLSpy (available at http://www.altova.com/) facilitates the management of large quantities of scripts. Another reason for choosing an XML scripting approach was due to the current inclination of IDWG (Intrusion Detection Work Group) of IETF (Internet Engineering Task Force) to use an XML based approach for IDMEF (Intrusion Detection Message Exchange Format)[11][2]. XML's intrinsic extensibility will be highly beneficial to our system in the long term.

6 Conclusion and Future Work

In this paper we have shown improvements made to the APHIDS system. We have pointed out some of the differences between the original APHIDS system [1] and the implemented system [8] which served as the starting point of our improvements. There were three main improvements made to the system: The first was the introduction of agent caching. We have shown that by caching agents instead of recreating them, we can benefit in terms of both latency and bandwidth. The second improvement was the introduction of a Policy Agent, which helps the Trigger Agent in making a more intelligent decision before reporting trigger events to the Correlation Agent. The third modification was the use of XML scripts to control our agent system. This was done mainly to make our DCS more user friendly by exploiting available XML visualization and editing tools.

An envisioned future improvement of APHIDS entails the provision of full support for multiple correlation servers.The use of a single correlation server introduces higher latency periods and is currently a potential bottleneck. Further, we are also investigating the use of Agent Cloning techniques for making our agents autonomous and our system autonomic.

[2] IDMEF is used as a common language for communication between different intrusion detection systems.

Acknowledgements

This research is supported partially by the Directorate of Telecom Engineering and Certification of the Department of Industry Canada. We would like to thank Ken Deeter, Christian Chita and Kapil Singh for leaving us with some valuable comments and ideas.

References

1. Deeter, K., Singh, K., Wilson, S., Fillipozzi, L., Vuong, S.: APHIDS: A Mobile Agent-Based Programmable Hybrid Intrusion Detection System. In: MATA 2004. (2004)
2. Roesch, M.: Snort - lightweight intrusion detection system for networks. In: Proceedings of USENIX LISA'99. (1999)
3. Balasubramanian, J., Garcia-Fernandez, J., Isacoff, D., Spafford, E., Zamboni, D.: An archi-tecture for intrusion detection using autonomous agents. Technical Report 98/05, COAST Laboratory, Purdue University (1998)
4. Asaka, M., Taguchi, A., Goto, S.: The implementation of IDA: An intrusion detection agent system. In: Proceedings of the 11th FIRST Conference. (1999)
5. Queiroz, J., Carmo, L., Pirmez, L.: Micael: An autonomous mobile agent system to protect new generation networked applications. In: 2nd Annual Workshop on Recent Advances in Intrusion Detection. (1999)
6. Kruegel, C., Toth, T.: Sparta - a mobile agent based intrusion detection system. In: Proceed-ings of the IFIP Conference on Network Security (I-NetSec). (2001)
7. Li, C., Song, Q., Zhang, C.: Ma-ids architecture for distributed intrusion detection using mobile agents. In: Proceedings of the 2nd International Conference on Information Technol-ogy for Application (ICITA 2004). (2004)
8. Deeter, K.: APHIDS: A Mobile Agent-Based Programmable Hybrid Intrusion Detection System. Masters thesis, Department of Computer Science, UBC. (2004)
9. Singh, K., Vuong, S.: Blaze: a mobile agent paradigm for VoIP Intrusion Detection Systems. In: Proceedings of ICETE 2004 - First International Conference on E-Business and Tele-communication Networks, Setubal, Portugal. (2004)
10. Cabri, G., Leonardi, L., Zambonelli, F.: XML Dataspaces for Mobile Agent Coordination. In: Symposium on Applied Computing (2000) 181–188
11. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format, draft-ietf-idwg-idmef-xml-14.txt. (2005)
12. Grasshopper. http://www.grasshopper.de. (2003)

Challenges in Modeling and Disseminating Context Information in Ambient Networks

Dineshbalu Balakrishnan[1], May El Barachi[2], Ahmed Karmouch[1], and Roch Glitho[2]

[1] MMARL, SITE, University of Ottawa,
161 Louis Pasteur St., Ottawa, ON, Canada - K1N 6N5
{dbalakri, karmouch}@site.uottawa.ca
[2] Concordia University,
1455 De Maisonneuve blvd. West, Montreal, Canada – H3G 1M8
{elbar_m, glitho}@ece.concordia.ca

Abstract. The concept of Ambient Networks (AN) aims at enabling the co-operation between heterogeneous networks on demand, in order to provide ubiquitous connectivity to end-users. This could be achieved by seamlessly incorporating context-awareness which plays a vital role in supporting autonomic decision making in ANs, while giving rise to new types of applications and services. Modeling and disseminating contextual information in a rapidly changing time-constrained heterogeneous environment, transparently to the potential users, is a challenge. This paper discusses these complications with an evaluation of the existing context modeling and dissemination approaches, with respect to ambient networking requirements.

1 Introduction

Ambient networking aims at providing an internetworking solution for the integration of heterogeneous networks, based on the dynamic composition of those networks [1]. This integration is translated in the guarantee of service continuity for the end-user, enabling him/her to use any device to access services seamlessly across any type of network. This is achieved through a set of universal control functions used as an overlay control plan to seamlessly integrate any network: the Ambient Control Space (ACS). In order to enable such dynamic behavior, decision making capabilities must be implemented in support of each function of the ACS. Context-awareness is expected to play a vital role in supporting autonomic decision making, providing by the same token an efficient approach for personalizing network services.

We have previously introduced a novel network-centric approach for context management in ANs [2] and highlighted its associated challenges. This approach aims at making the network itself context-aware by collecting and making use of the surrounding contextual information for network operations and decision making. This contextual information is not only used by the network entities (i.e., network services) but can also be provided to applications/ end-user services. Fig. 1 illustrates this network-centric approach.

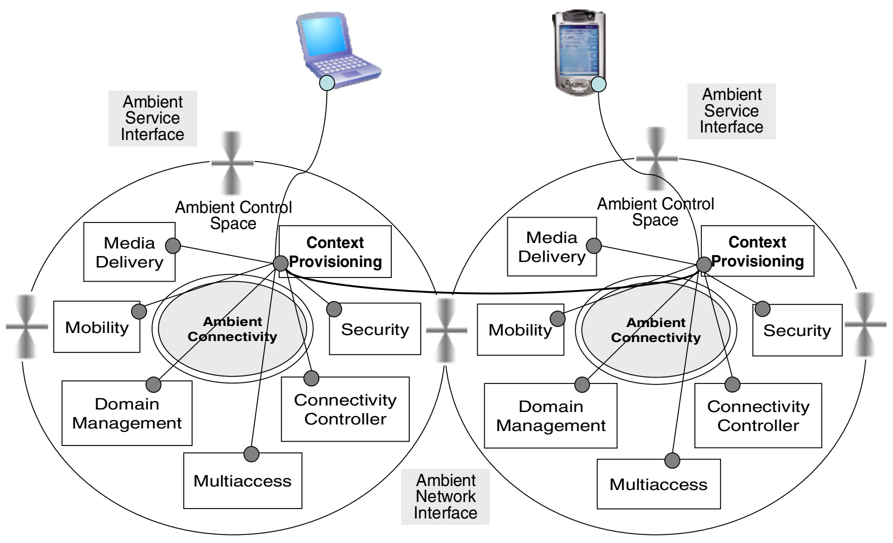


Fig. 1. The network-centric context management approach used in ANs

Context management involves several complicated tasks such as collecting the raw contextual data from various sources (context sensing using CSC[1]), processing and modeling of this data, and finally disseminating the refined information which could be used by a context aware system/ service (CAS) for its operation or adaptation. This paper focuses on the modeling and dissemination tasks. The dynamics and heterogeneity envisaged in ANs accentuate the need for efficient context modeling and dissemination. On one hand, a well-designed *context information model* is needed to efficiently utilize and share the wealth of information available. On the other hand, a suitable *context dissemination strategy* is critical to ensure the availability of up-to-date contextual information, in a scalable manner. Although these tasks were studied under various circumstances and for different types of networks, their execution from an AN point of view is little known and worth examining.

This paper focuses on evaluating the current approaches related to these two issues, and highlights the research challenges associated with modeling and disseminating context information in ANs. We first discuss modeling and evaluate the existing modeling approaches, in section 2. In section 3, we tackle the issue of context dissemination. We finally conclude the paper in section 4.

2 Modeling Context Information in Ambient Networks

In order to support context-awareness, there is a need for developing a formal context model to facilitate context representation, sharing, and semantic interoperability of

[1] In Context Sensitive Communications, the communication channel is established between a pair of devices (for e.g., network nodes or mobile devices) based on some specific contexts (for e.g., time or bandwidth).

heterogeneous systems. In this section, we start by listing the requirements imposed by ambient networking on the context modelling approach to be used. We then evaluate the current context modelling approaches and discuss the open research issues.

2.1 Requirements

ANs aim at expanding the existing network capabilities in the areas of network composition, mobility, and heterogeneity. In order to support these features, we believe that the adopted modeling approach should satisfy the following requirements:

1. *Formality*: It is crucial that all entities within the network share the same interpretation of the information exchanged. This can only be achieved by a formal (i.e., precise and unambiguous) description of this information.
2. *Support for efficient reasoning and inference mechanisms:* E Efficient reasoning mechanisms are needed for deriving higher-level context information from lower-level pieces of information; for evaluating context queries; for model checking; and for reasoning about relationships between different models.
3. *Support for fast access to contextual information*: The ability to rapidly retrieve the needed information is critical in the dynamic environment offered by ANs.
4. *Support for imperfect contextual information:* Context information, by its nature can be uncertain, incomplete, or unavailable at times. This aspect should be covered, for instance by interpolation of incomplete data on the instance level.
5. *Support for information quality indication:* Context information can be of varying quality and richness, depending on its source. The context information model should therefore support quality indications.
6. *Support for dynamic creation, modification, and extensions of the model:* ANs are formed dynamically of collaborating heterogeneous networks, therefore requiring the capability for dynamic creation and modification of model instances. Furthermore, the model should be easily extensible.
7. *Support for models merging:* The composition of different networks requires the capability to dynamically merge their information models. This implies solving any inconsistencies/contradictions between the model's concepts.
8. *Support for models checking and partial knowledge validation:* As ANs evolve, the ability to perform model checking in order to detect any errors in the newly created models becomes important. It is also essential to be able to validate context on structure and instance levels, against an existing model.
9. *Support for interoperability between different models:* Mechanisms should be developed to enable interoperability with models used by legacy systems.
10. *Applicability to existing service environments:* From the implementation point of view, the context model should be applicable within the existing infrastructures envisaged for ambient computing.

2.2 Context Modeling Approaches

Key-value pair models: The key-value pair model is the simplest data structure for modeling context. It was first proposed by Schilit et al. [3], then used in other projects such as Mobisaic [4]. This model uses an environmental variable acting as the key and the value of the variable holding the actual context data. This approach allows for pairs recursion and is powerful enough to allow pattern-matching queries.

Markup scheme models: These models use a hierarchical data structure in which contexts are modeled as tags and corresponding fields. The fields can recursively contain other tags and corresponding fields. There is possibility for generating triggers based on conditions specified in some tags. This approach is typically used for the modeling of profiles since it naturally reflects the structure of profile information. Such models are usually based on a derivative of SGML (standard generic markup language – a superset of XML). Comprehensive Structured Context Profiles (CSCP) [5] use this modeling approach. This framework provides a flexible overriding/merging mechanism, allowing to override and/or merge a whole profile sub-tree.

Graphical models: A popular general purpose graphical modeling instrument is the Unified Modeling Language (UML). Due to its generic structure, UML can also be used to model context. In fact, UML extensions have been previously used to model contextual aspects relevant to air traffic management. Another example is the context modelling language (CML) [6]. CML provides a graphical notation for describing types of information (in terms of fact types), their sources, specify dependencies and constraints, as well as quality labels. Models developed using this approach can be easily mapped to relational databases.

Object-oriented models: These models inherit from the object-oriented paradigm the ideas of encapsulation and reusability. In such models, the details of data collection and processing is encapsulated on an object level and therefore hidden to other components. Access to information is provided through specified interfaces. The active object model of the GUIDE project [7] is a representative of this approach.

Logic-based models: Logic-based models are highly formal models which express contextual information as facts in a rule-based system. In such system, a set of rules define the conditions that should be satisfied in order for a certain conclusion to be reached . Although logics are sometimes used for context modelling (such as in the sensed context model [8]), they are mainly used in conjunction with other modeling approaches as context reasoning mechanisms. For instance, Henricksen et al. use a variant of predicate logic to derive higher level contextual information [6].

Ontology based models: An ontology is a formal explicit description of the concepts in a domain, the properties of each concept, and restrictions on those properties. By defining the common vocabulary that entities need to share information about a certain domain, ontologies become promising tools for enabling knowledge sharing and reuse. More recently, the increasing popularity of ontologies has led to the appearance new ontology-based models for context. These models aim to better support interoperability (between entities sharing common concepts), and leverage the existing tools for ontology definition and checking. Such models also have the potential to support sophisticated ontology-based reasoning. The Context Ontology (CONON) [9] is an example of ontology-based context models.

Hybrid models: Recently, some research efforts have been carried to investigate the benefits of using a hybrid modeling approach. For instance, Henricksen et al. [10]

investigated the incorporation of ontology-based reasoning to their graphical modeling approach, mainly to achieve model checking and reasoning about models interoperability. Becker et al. [11] also examined the benefits of combining spatial context models (built upon data modeling techniques) and ontology-based context models.

2.3 Evaluation and Open Research Issues

We now evaluate the presented approaches with respect to the requirements derived, and discuss the open research issues. Table 1 presents a summary of the evaluation.

Key-value pair models use an extremely simple modeling technique which lacks formality and sophisticated structuring capabilities. On the other hand, fast information retrieval is achieved using pattern-matching queries. Other strengths of this approach are the efficient and scalable information management.

Markup scheme models reply on standardized markup languages, therefore achieving a high level of formality and applicability in markup-centric environments. The associated markup query languages allow for fast information retrieval. Quality indicators are supported, and overriding/merging mechanisms allow for dynamic modifications/ extensions to the model. However, this approach shows deficiencies in terms of reasoning capabilities, incompleteness, and models merging / interoperability.

Graphical based models are mapped onto relational databases, allowing database queries for fast information retrieval. Logic-based reasoning has been used with such models. Quality and incompleteness meta-data can also be incorporated. Dynamic modification/extension of the model is subject to some constraints on the structure level. Although partial validation is possible, model checking and models interoperability cannot be achieved. Finally, these models offer a lower degree of formality.

Object-oriented models exhibit a certain degree of formality through the use of well-defined interfaces. Incompleteness and quality parameters can be associated to the information and the models can be easily modified / extended, and checked. Partial validation is also easily achievable. On the other hand, models merging and models interoperability are not supported. Object-oriented models are widely applicable to the existing object-oriented environments.

Logic-based models offer a high degree of formality, and powerful reasoning mechanisms. Information incompleteness, information quality, and models extensibility are addressed. The weaknesses of this approach include its lack of support for models merging and its limited applicability to the existing environments. In fact, such models require the availability of logic reasoners, and impose strong requirements on the resources of the computing devices.

Beside their high degree of formality, *ontology-based models* exhibit strengths in terms of semantic expressiveness and the powerful reasoning capabilities they offer. On the other hand, the complexity of the reasoning mechanisms comes with a price in terms of performance, as shown in [9]. Beside the performance aspect, ontology-based models are very strong on all remaining requirements. In terms of applicability, they require environments capable of handling ontology description languages.

Table 1. Evaluation of the existing context modeling approaches

Requirement/ Approach	Key-value	Markup	Graphical	Object oriented	Logic-based	Ontology-based
Formality	-	+	-	+	++	++
Reasoning	-	-	+	-	++	++
Fast info retrieval	Linear -fast	Markup QL -fast	DB query -fast	Methods calls -avg	Reasoning (depends on complexity)	Reasoning (depends on complexity)
Imperfection	-	-	+	+	+	+
Quality	-	+	+	+	+	+
Dynamic model / extension	-	+	-	++	++	++
Models merging	-	-	-	-	-	+
Checking & Part. validation	-	++	+	+	-	++
Interoperability	-	-	-	-	-	+
Applicability	+	++	+	+	-	+

Work on *hybrid models* is still at an early stage. Therefore, we did not find enough details to evaluate this approach. Nevertheless, we certainly find these efforts interesting as they have the potential of combining efficient information management with powerful reasoning mechanisms.

In light of this evaluation, we conclude that ontology-based modeling seems to be the most promising approach with respect to our requirements. A hybrid modeling approach also seems to be very promising, although not enough work has been done to fully evaluate it. In order to build an ontology-based context model for ANs, we must face the challenge of controlling the scale of the dataset used, while still covering the large variety of information handled. In fact, In order to produce a scalable solution, specific ontologies related to different domains should be developed, instead of a monolithic general ontology. We should also try to minimize the complexity of the rule set used in order to achieve a satisfactory performance. Finally, if the performance issues cannot be surmounted, a hybrid approach should be considered.

3 Disseminating Context Information in Ambient Networks

Context dissemination deals with spontaneously propagating the modeled contextual information from sources to sinks. The process's facilitation in ANs and context-aware ambient applications is significant to maintain scalability and dynamicity [2, 1]. This section discusses the challenges associated to context dissemination, with an evaluation of existing context dissemination approaches with respect to ANs. An overview of the process of context dissemination is illustrated in Fig. 2.

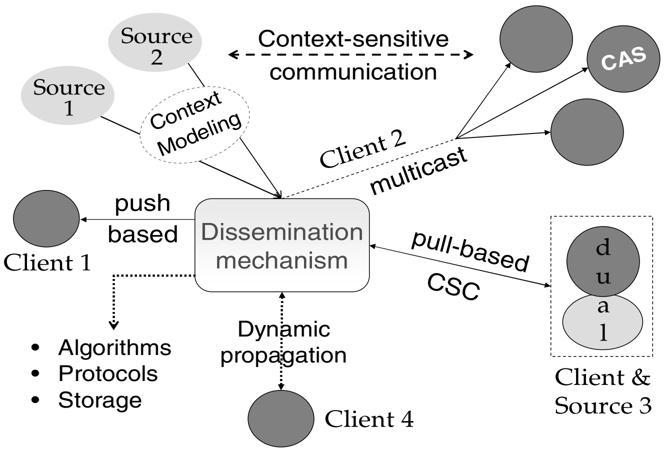


Fig. 2. Overview of the context dissemination process

3.1 Requirements

In order to support the ambient networking characteristics, the following requirements should be satisfied by the espoused context dissemination approach:

1. *Support for Adaptation*: To guarantee continuous service delivery to clients, the dissemination mechanism should be correct and adaptable by supporting efficient inference mechanisms based on the modeling scheme employed and by extending processes via merging of dissemination strategies.
2. *Quality of Context*: As the dissemination protocols should provide solutions for scalability and correctness, the Quality of Context (QoC) used should be precise.
3. *Support for Interoperability*: Mechanisms should be developed to enable interoperability between the dissemination protocols employed in different networks.
4. *Support for Negotiation-based Dynamicity*: The employed dissemination strategy should be capable of making dynamic decisions based on real-time negotiation of entity's contracts. The CLA negotiation protocol [12] illustrates this capability.
5. *Content-based Dissemination*: The process of content-based dissemination is based on the content being transmitted in the message. As the information flow in ANs is 'emergent' and as the client's attributes are themselves derived from the information content, the dissemination process should be content-based instead of channel- or subject-based.
6. *Delivery option and Periodicity*: Other customary dissemination requirements include the selection of appropriate delivery and periodicity options such as push- or pull-based and periodic or aperiodic, respectively.
7. *Privacy and Security*: The employed dissemination protocols should be secure enough in an ambient environment in addition to honouring privacy concerns.
8. *Spontaneity and Applicability*: The dissemination strategy should support spontaneous access to information and should be applicable in the existing infra structures.

3.2 Context Dissemination Approaches

Dissemination using a Middleware: The main reason for middleware employment for context dissemination is to maintain and guarantee data dissemination reliability in critical applications, in addition to reducing a system's overall burden. A context dissemination middleware flexibly routes the modeled contexts to appropriate heterogeneous receivers based on the content, channel, or subject. However, this approach is inadequate for ambient environments due to its high time consumption and mediocre dynamic characteristics. As it widely differs from the process of service delivery, the employed middleware isn't a typical proxy server or clustered backup of servers. The Information Flow Graph (IFG) model [13] uses this approach.

Content-based Dissemination using ENS: Context dissemination using an Event Notification Service (ENS) normally follows the 'publish-subscribe' mechanism (for e.g., Siena [14]). On the other hand, content-based systems offer more flexibility then the traditional publish-subscribe systems, as they allow clients to compose context requests based on the content of published events [15]. Event generators publish notifications to the infrastructure which sends them to event subscribers. Herein, the infrastructure's main functions include notification selection & dissemination. Notification dissemination can be performed using event-servers, shortest-path matching & forwarding algorithms, and network-level multicast communication techniques. In the case of distributed ENSs, servers are interconnected, each serving a subset of the clients [14].

Graph-based Context Dissemination: Graph-based abstractions are used to perform dissemination in popular projects such as the IFG [13] and Solar [16]. In such systems, a mediator receives contexts from different sources and disseminates (push-based) the current context information and changes to appropriate applications. In p-to-p, information is delivered to a data client, while in 1-to-n, either broadcast or multicast is used. In the latter mode, scalability can be increased by the use of an operator graph. An operator is an object which accepts subscriptions (event streams) as input, processes them, and finally disseminates a new event stream. An operator graph is a directed acyclic graph of operators formed from the combination of applications' subscription trees. While this approach offers advantages in terms of flexibility and scalability, it also poses the challenge of performing context-sensitive dissemination from a publisher to multiple subscribers.

Content-based Routing using XML: To facilitate content routing over mesh-based overlay networks to heterogeneous clients, application-level XML routers are used to forward data streams in the form of XML packets (such as in reference [17]). These routers disseminate individual packets either to clients or to other routers. This approach is more reliable and dynamic than tree-based networks as the distributed network of XML routers stream real-time contexts. Hence, this approach aids to achieve context dissemination in varying heterogeneous domains.

3.3 Evaluation and Open Research Issues

Consider a generic scenario wherein a user with his/her own PAN is decomposing from his/her home network and dynamically composes with a car network in the vicinity (in a transparent fashion). With this type of scenario as the background, the dissemination approaches discussed above are evaluated against the requirements discussed in section 3.1. The results are summarized in table 2 followed by the open research issues and further recommendations discussion.

Table 2. Evaluation of context dissemination approaches

Approach / **Properties**	Middleware-based	Content-based using ENS	Graph-oriented	Content-based using XML
Adaptability	++	+	++	-
QoC	++	+	+	+
Interoperability	-	-	-	-
Dynamicity / Negotiation-based	-	-	-	+
Content-based	+	++	+	++
Delivery option	hybrid	push-based	push-based	push-based
Periodicity	hybrid	aperiodic	aperiodic	aperiodic
Privacy & Secu-rity	-	-	-	-
Spontaneity	-	+	-	+
Applicability	+	+	+	+

- While the support for adaptation is good in both middleware- and graph-based systems, it's considerably low in content-based systems. Distributed ENSs offer a better adaptability due to their inference mechanisms and their extensible nature.
- The quality parameters in context-aware networks are associated to the acquired contexts in all the approaches.
- The interoperability and negotiation-based dynamicity properties aren't yet entirely handled in all the approaches; though the distributed network of XML routers partially supports dynamic context dissemination. The middleware-based approach for instance is time-consuming and its dynamicity characteristics are inadequate.
- The concept of content-based dissemination, though complex and time-consuming from an AN point of view, is considered in all dissemination strategies.
- Although spontaneity and applicability are moderately acceptable in the surveyed context dissemination approaches, privacy and security aren't handled.

With respect to the evaluation we conclude that the demand for advanced dissemination mechanisms for complex context-aware services is immense. As shown in table 2, none of the surveyed approach's applicability is entirely satisfactory. Though content-based dissemination approaches perform moderately, there exists no formal method to enhance the processing strategy. The challenging push-based distributed dissemination needs to be widely employed unlike context-aware middleware architectures, while the QoC precision and negotiation-based dynamicity are crucial too. Finally, in order to surmount the performance issues of context dissemination approaches in ANs, novel interoperable dissemination protocols are required. We suggest the employment and enhancement of CLA language and negotiation protocol (see [12]), along with ontology-based content-reduced policy profiles [18].

4 Conclusion

With the dynamics and heterogeneity envisioned in ANs, context-awareness is a key feature. Some of the challenges associated with context management include the development of a suitable context information model and the elaboration of an efficient information dissemination strategy. In this paper, we derived requirements imposed by ambient networking on the modeling and dissemination approaches to be used, and evaluated the existing approaches with respect to these requirements. From the evaluation of current context modeling approaches, hybrid and ontology-based models seem promising. On the other hand, the evaluation of existing context dissemination approaches suggests the need for novel dissemination approaches based on CLAs, ontologies, and policy-profiles.

Acknowledgement

This paper describes work undertaken in the context of the Ambient Networks - Information Society Technologies (IST) 507134 project. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the Ambient Networks project or the Context Aware Networks workpackage. This work is partly funded by an NSERC SRO grant.

References

1. N. Niebert, et. al.: Ambient Networks -- An Architecture for Communication Networks Beyond 3G, IEEE Wireless Communications (2004)
2. A. Karmouch, et. al.: Contextware Research Challenges in Ambient Networks, IEEE MATA 2004 – 1st International Workshop on Mobility Aware Technologies and Applications, Florianopolis, Brazil (2004)
3. B. N. Schilit, M. M. Theimer, and B. B. Welch: Customizing mobile applications, Proc. of USENIX Mobile and Location-Independent Computing Symposium (1993) 129-138
4. G. Voelker and B. Bershad, Mobisaic: An information system for mobile wireless computing environment, In: Proc. of IEEE Workshop on Mobile Computing Systems and Applications (1994) 185-190
5. A. Held, S. Buchholz, and A. Schill: Modeling of context information for pervasive computing applications, In: Proc. of the 6th World Multiconference on Systemics, Cybernetics, and Informatics (2002)
6. K. Henricksen, J. Indulska, and A. Rakotonirainy: Modeling context information in pervasive computing systems, In: Proc. of the 1st International Conference on Pervasive Computing (2002) 167-180
7. N. Davies, K. Cheverst, K. Mitchell, and A. Friday: Caches in the air: Disseminating tourist information in the GUIDE system, In: Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (1999) 11-19
8. P. Gray, and D. Salber: Modelling and using sensed context information in the design of interactive applications, In: Proc. of the 8th International Conference on Engineering for Human-Computer Interaction (2001)

9. X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung: Ontology based context modeling and reasoning using OWL, In: Proc. of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (2004) 18-22
10. K. Henricksen, S. Livingstone, and J. Indulska: Towards a hybrid approach to context modelling, reasoning and interoperation, In: Proc. of the 1st International Workshop on Advanced Context Modeling, Reasoning, and Management (2004)
11. C. Becker, and D. Nicklas: Where do spatial context-models end and where do Ontologies start? A proposal of a combined approach, In: Proc. of the 1st International Workshop on Advanced Context Modeling, Reasoning, and Management (2004)
12. M. Khedr and A. Karmouch: A Semantic Approach for Negotiating Context Information in Context Aware Systems, IEEE Intelligent Systems, to be published (2005)
13. G. Banavar, et al.: Information Flow Based Event Distribution Middleware, In: Proc. Of the 19th IEEE Intl. Conf. on Distributed Computing Systems (1999)
14. A. Carzaniga, D.S. Rosenblum, and A.L. Wolf: Design and evaluation of a wide-area event notification service, ACM Trans. Comput. Syst., vol. 19 (2001) 332–383
15. G. Chen and D. Kotz: Supporting adaptive ubiquitous applications with the SOLAR system, tech. report TR2001-397, Dept. of Computer Science, Dartmouth College (2001)
16. G. Chen and D. Kotz: Solar: A pervasive-computing infrastructure for context-aware mobile applications, tech. report TR2002-421, Dartmouth College (2002)
17. Alex C. Snoeren, Kenneth Conley, and David K. Gifford: Mesh-Based Content Routing using XML, In: Proc. of the 18th ACM Symposium on Operating Systems Principles, Banff, Canada (2001)
18. Guanling Chen and David Kotz: Application-Controlled Loss-Tolerant Data Dissemination, tech. report TR2004-488, Dartmouth College (2004)

A Co-designed Hardware/Software Architecture for Augmented Materials

Simon Dobson[1], Kieran Delaney[2], Kafil Mahmood Razeeb[3], and Sergey Tsvetkov[4]

[1] Department of Computer Science, University College, Dublin IE
simon.dobson@ucd.ie
[2] Cork Institute of Technology, Cork IE
kieran.delaney@cit.ie
[3] Tyndall National Institute, Cork IE
kmahmood@tyndall.ie
[4] Department of Computer Science, Trinity College, Dublin IE
sergey.tsvetkov@cs.tcd.ie

Abstract. Recent advances in materials, sensing, power harvesting, context-awareness and miniaturisation have opened-up the possibility of constructing materials that directly include considerable computing power. We present an architecture for the hardware/software co-design of such "augmented" materials that allows designers to address the links between the physical and informational properties of artefacts. The architecture is highly modular in terms of sensor, communications, power and processing capabilities, and utilises an advanced semantically well-founded software model.

1 Introduction

The inner workings of today's technologies (and those of any era) are always notionally complex to everyday users, and the same can be said of objects made from such technologies. The process from which these objects are made to appear intuitive and simple hides a multitude of technological and conceptual development cycles, reaching a stage where their capabilities are well-proven, trustworthy, easily understood, and effectively taken for granted.

Recent advances in materials technology, sensor and processor miniaturisation, power harvesting and context-aware software have opened-up the possibility of constructing "augmented" materials that include computational intelligence and communications directly into the fabric of artefacts. These materials can be used for a range of applications where designers need to combine physical and informational effects very closely. By co-designing the physical, hardware and software aspects of such materials it is possible to leverage the strengths (and address the weaknesses) of the individual components to achieve a seamless pervasive technology.

Our focus on materials is driven by four observations:

1. the useful behaviour of an object is a function of the capabilities of its component materials, both physical and informational;
2. many behaviours depend both on the geometric shape adopted by the object; and its relationships with other objects;
3. complex behaviours and interactions are more easily understood if composed from discrete and recognisable objects; and
4. the close correspondence between physical affordances and information effects is what allows simple, scrutable pervasive computing.

We believe that – by broadening what we mean by the "behaviour" of materials to include sensing, processing and communications, as well as physical, aspects – it is possible to develop an architecture for co-designed pervasive computing systems expressed in terms of material interactions. Materials are infused with systems capability that allows a digital representation to be developed at a selected formation stage (*e.g.* curing) and maintained thereafter. To pursue the analogy, an augmented material is an alloy of a traditional material and a processing capability that yields behaviours different from those of its individual components. An effective implementation yields a situation where any subsequent materials processing capability behaves as a programming step, affecting the digital as well as physical characteristics of the material.

In this paper we describe an architecture for augmented materials. The architecture is highly modular in terms of sensor, communications, power and processing capabilities, and utilises an advanced semantically well-founded software model. By integrating materials properties, sensing and semantics we aim to deliver a robust and flexible platform for self-managing, autonomic smart systems and artefacts.

Section 2 motivates our work and places it into the context of pervasive computing research. Sections 3 and 4 respectively describe the hardware and software components of the architecture, with particular emphasis on their complementarity. Section 5 contrasts our approach with other work in the field, especially in wireless sensor networks, while section 6 concludes with our immediate future directions.

2 Design Space and Issues

Materials science has expanded greatly in the past few years with contributions from nanotechnology as well as from traditional physics. However, even the smartest of "smart" materials lacks the flexibility and sophistication of software. Augmenting materials with information technology addresses this gap, allowing materials to process information as well as functioning physically, reflecting on its behaviour and providing feedback to the wider IT environment.

An augmented material might be used as part of a rigid structure such as an airframe. With embedded sensing the material can detect changes in its physical composition. It can provide ground crew with an on-going report on its health or, in case of damage or failure in-flight, inform the flight control systems to take remedial action such as reducing the manœvering envelope to avoid excessive strain. These observations may be performed using piezo-electric sensors, and are used externally on some airframes: internalising the sensing is more robust and decreases the load on the central computers. Similar techniques can be applied to other built structures such as buildings, bridges and ships.

At a more personal level, an increasing number of people undergo physiotherapy to accelerate the healing of damaged bones. A major problem occurs in monitoring a treatment regime to ensure that the patient performs the necessary exercises but does not over-tax the injury. A augmented cast could monitor the impact on the supported limb to ensure that it remains within acceptable limits, providing feedback to the patient and therapist. A material whose stiffness is variable could actively modify its physical characteristics according to a complex scheme encoded in software.

Truly pervasive computing applications are limited by the availability of sufficiently small devices on which to run applications. Typically one encounters smart-building systems in which artefacts are tagged with RFID and tracked by computers in the walls. This is an asymmetric solution, as the artefact cannot perform actions autonomously. By constructing artefacts with augmented materials the symmetry is restored, allowing us to build (for example) books that know their location and relationships to other artefacts. This is a more peer-to-peer solution than is possible at present, and relies critically on the ability to place processing alongside sensing in the fabric of everyday things.

There is a body of work on sensor miniaturisation, sensor networking, pervasive and amorphous computing (section 5). What makes the current context novel is the application of these technologies specifically to new materials from which to construct objects, rather than as addenda to existing objects.

At the physical level, an augmented material consists of the substrate material and a (possibly extremely large) collection of embedded processor elements with local sensing and communications. The choice of substrate governs the gross physical properties of the augmented material, and so is conditioned by the final application. Possible choices include flexible and non-flexible plastics, fibre-glass, fabrics, concrete and metals – indeed *any* material whose application would benefit from augmentation with IT.

The added value of an augmented material comes from its additional interaction capabilities, which in turn derive from the software it can support. Such materials offer significantly different characteristics to more traditional embedded systems, both in terms of the number of elements involved (potentially thousands) and the unusually close relationship between the elements and the material in which they are embedded.

We may view an artefact constructed from an augmented material at four distinct levels.At the **physical** level, the material exhibits certain structural properties such as stiffness, ductility, conductivity and so forth, which condition the physical applications for which it can be used. At the **element** level, each processing element in the material functions as an independent component capable of local sensing, local processing and communications with nearby elements. At the **material** level the individual elements co-ordinate their actions to present a global behaviour, typically integrating the local sensing information into a global view of the material. At the **artefact** level the material can "understand" its function and relationships with other augmented materials in its vicinity. An augmented-material book lying on an augmented-material table provides a good illustration of the four levels of concern.

This model also illustrates the challenges facing augmented materials. Although each individual element may provide local sensing and processing, element-level observations are of little use in isolation: they need to be integrated at the material level. Even given this integration, the significance of the various observations can only be determined by using quite sophisticated reasoning informed by a knowledge of the real-world environment (artefact level). At the element level we need to be able to drive the available sensors, perform calculations and communicate with other elements. These individual observations must be combined at the material level to provide an integrated view of the material, and used to drive inferencing at the artefact level. None of these problems is amenable to direct programming solutions of the kind normally found in embedded systems.

3 Sensing and Processing

These considerations lead us to a *co-designed* architecture in which we consider the hardware (material, sensing, processing, communication) and software (knowledge, reasoning, task) aspects together. The various properties of augmented materials can then be traded-off against each other to provide a self-configuring and robust information platform.

Making a material self-aware involved two stages. At the first stage (curing), the elements establish a "baseline" view of their configuration. The second stage (lifetime) updates this representation over the material's lifetime. As an example, consider a flexible sheet of plastic containing elements with local strain sensors. At the first stage it establishes its initial shape (flat) and the relative locations of its elements; during its lifetime it updates its view of its own shape by integrating the local strain observations of the elements. Such a material "knows" its own shape on an on-going basis, and this can be used to inform its more advanced behaviours.

3.1 Communications and Location

There are three aspects of the communication for augmented materials: between sensors and processors, between elements and between materials. Sensor/processor communications can be performed locally by integrating them onto the same device. Inter-material communications can occur either wirelessly when elements at the edge of one material come into range of elements at the edge of another, or through wires if materials are plugged together or connected to the wider population of internet servers.

Inter-element communication is perhaps the most interesting. Communications may be point-to-point or broadcast; point-to-point will typically use wires run through the matrix; broadcast may use radio, conduction through the substrate, or even acoustics. The point is to match the communications requirements to the properties of the substrate and the target application.

Since elements are returning information about their locality, many applications will require that they know their position in the material matrix in some co-ordinate system in order to correlate observations with locations. It is generally possible to identify element locations at fabrication time, for example placing elements in a uniform grid, and then informing each element of its location at the curing stage. While this is the simplest solution, it makes fabrication more difficult and may introduce errors if elements drift during processing.

A more attractive solution from a fabrication perspective is to locate elements randomly within the substrate and have them work out their location post-fabrication. Locating an element in this way is an interesting problem in its own right, but that we believe self-location of nodes with wireless communication to be feasible given assumptions of a sufficiently dense network with a sufficient number of elements, suitably distributed, whose locations are known.

3.2 Current State

We have designed and constructed elements for use within a range of substrates across the design space described above. Our design is highly modular, allowing individual components to be changed within the overall design framework. Modularisation includes communications, sensing, processing, memory and power source. It allows us to (for example) change the communications technology independently of the sensors, or to deploy additional sensors at the expense of memory or processing power.

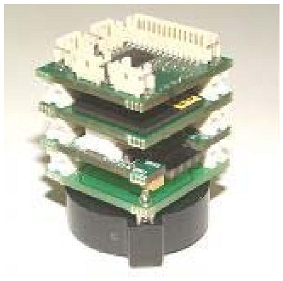
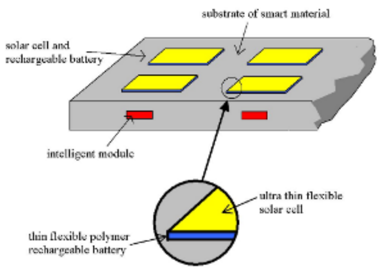


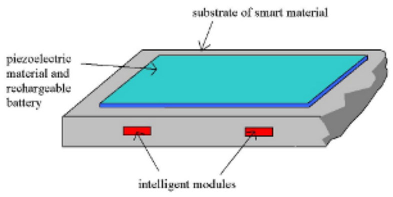
Fig. 1. 25mm prototype module with coin battery

Our current hardware platform[1, 2] consists of a number of 25mm-on-a-side FPGA-based elements (using Xilinx Spartan IIE cores) placed in formation without a substrate material (figure 1). We are currently evaluating the sensing, power management and inferencing capabilities of this platform, using the results to inform the design of the next steps on the road-map: 10mm and 5mm elements, embedded in a flexible plastic.

The communications transceiver consists of a fully integrated frequency synthesiser, a power amplifier, crystal oscillator, modulator and antenna. Output power and frequency channels are easily programmable. Current consumption is very low, and a built-in power-down modes makes power-saving easily realisable. The designer can easily program the device with custom protocols. The element's sensor array connects though a dual-channel RS232 transceiver. The module contains two A-to-D converters for interfacing up to seven external


substrate of smart material
solar cell and rechargeable battery
intelligent module
ultra thin flexible solar cell
thin flexible polymer rechargeable battery

(a) Solar cell array


substrate of smart material
piezoelectric material and rechargeable battery
intelligent modules

(b) Piezoelectric battery charging

Fig. 2. Possible power harvesting approaches

sensors that change their resistance according to the parameter being measured. The modules use a stackable connector system to make the electrical and mechanical interconnections between themselves.

Power supply is one of the major obstacle towards the miniaturisation of the autonomous sensor nodes. Coin cells may be used to provide power to the system, connecting directly using the stackable connector system, but power harvesting is obviously a much more attractive option for a long-lived, autonomous material. Figure 2 shows two examples of a smart material utilizing an array of ultra-thin flexible solar cells as the energy harvesting power source and an flexible polymer rechargeable battery for energy storage (figure 2(a)), suitable for outdoor use, and an alternative piezoelectric crystal that can generate small currents from vibrations to charge the rechargeable batteries (figure 2(b)). These are laminated onto the augmented material.

4 Programmability

Individual elements present a diverse set of possibilities. Although each element shares a common microcontroller core (and so can potentially run the same software), the population of sensors, actuators and other devices can vary widely. Open-ended behaviour means that augmented materials are not amenable to direct programming solutions of the kind normally found in embedded systems, so we have adopted a less familiar (but more powerful) approach based around rich, scalable, self-organising context models and inference.

Our overall goal is to integrate programming, as far as possible, into the process of manufacturing augmented materials, and to capture clearly the relationship between factors affecting the material and their behavioural effects[3].

4.1 Programming Elements

We took the decision to mirror the structure of the hardware module in the software platform. The initial population of software components mirrors the design choices made in hardware. As well as simplifying configuration, this approach allows us to make modules "reflective", in the sense that they "know" their sensing and other capabilities from the outset.

In the wider field of pervasive computing, many systems have adopted a more knowledge-based approach, modeling context as a collection of assertions (a good example is [4]). Typically this involves combining a basic knowledge representation with an ontology management system for expressing the constraints on knowledge. Given the constraints of power and space we are working under with augmented materials we have chosen to use RDF[5] for representing context, decoupling the data model from its usual XML format to compress the knowledge base sufficiently to operate on a microcontroller. RDF structures knowledge in terms of $(subject, predicate, object)$ triples. Each triple captures a single binary relationship (represented by the predicate) between two entities (the subject and the object). The collection of triples can be read as a concept graph, with each triple defining a labelled edge. The available predicates for a vocabulary for talking about a particular sort of knowledge, with the relationships between the predicates being structured by an ontology. The subjects are the "denotable values" in the knowledge domain, typically either identifiers for things or literal values. Predicates are also URLs, allowing unique labeling of predicates within an ontology without a centralised registry.

Each sensor on an element has an associated vocabulary providing predicates for each type of sense data. A strain sensor, for example, might define predicates for the strain in newtons and the direction of strain in some co-ordinate system. For each sensor, the device driver maps the sense data into the local context model using this representation (figure 3).

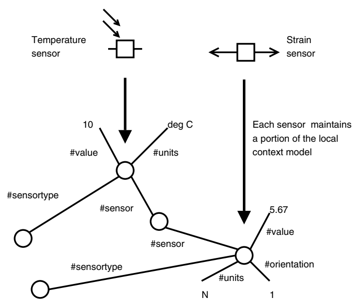


Fig. 3. An individual model held on an element

The advantage of this approach is three-fold. Firstly, it provides a common framework within which to represent and query *all* sensed data for any client that knows the predicates. Secondly, it allows the same information to be sensed by different sensors in a manner that is largely transparent to the programmer, simply by sharing vocabularies. Thirdly, it raises the abstraction level for programmers away from hardware and into the knowledge plane, making it easier to express logical constraints and queries.

4.2 Programming Materials

A single augmented material might include several hundred sensor and processing elements. Current techniques to embedded systems stress programming the *devices* as the basis for applications, which will not scale up to such environments. Instead we need to program *materials as a whole.*

Since we are focusing on augmented materials, we are dealing with a more limited application domain than might be found in general amorphous computing[6]. Specifically, we assume that applications are primarily concerned with reacting to the *physical environment* of the material and its *physical and informational properties.* This establishes a close correspondence between the material and the elements within it: the location of an element within the substrate will typically be a significant factor in interpreting its information.

Given a located element, its sensor observations can be related directly to the environment and/or operation of the augmented material. In one sense integrating sense data is simple: any element wanting to know the state queries the model on each appropriate element and performs the necessary numerical integration. In a system with low-power unreliable elements and local communications, however, things are not so simple.

A naïve solution would nominate a single master element to maintain a global view of the material. Such a node would require significant storage and computing power and would be a single point of failure for the material. At the other end of the spectrum, nodes would exchange their local states with all other nodes *via* a gossiping protocol[7] .Gossip protocols are used extensively in *ad hoc* allowing more queries to be answered using only local knowledge.

While a fully decentralised implementation has its attractions, distributing even aggregated data from the entire material across all elements greatly increases their computational, storage, communication and (most importantly) power requirements. Locality means that elements do not in general need to know results from remote elements in the material (although they may need to be able to find out special cases). We believe that a hybrid approach offers a good trade-off between decentralisation and efficiency. Elements are divided into two categories – sensing elements and aggregating elements – which are then evenly distributed through the substrate. Aggregating elements can have more storage capacity than sensing elements.

The two classes gossip, but in different ways. Sensor elements gossip with nearby aggregating elements by sending changes in their local states, which may then be aggregated to provide a summary of the state of the local area. The intention is to provide explicit storage and computational capacity for these summarising activities. By summarising locally the system scales well as the material grows (assuming a "fair mix" of elements) and requires only local communications.

Aggregate elements gossip with other aggregate elements, but exchange management information about which aggregate is summarising what locale. The protocol essentially distributes the set of subjects and predicates that an aggregate is holding, providing an RDF-level summary of the information stored on the element. When a query is made to an aggregating element, it can determine which elements can have information that may satisfy the query and perform the appropriate sub-queries. In effect the material behaves as a single RDF model that can be queried as a whole, with queries being decomposed and distributed as required. This structure means that data is held locally within the material but accessed globally. The overall information content of a material is available *via* any aggregate element using an identical interface, regardless of the actual location of the information in the substrate.

4.3 Internal *Versus* External Semantics

While the low-level sensor data can be managed largely automatically, being tied closely to the physical realisation of the material, inferred information is handled using rules provided by the programmer within a truth-maintenance framework in which changes in lower-level information propagate to higher-level views inferred from that information.

However, it is important to realise the way in which sense data – the “internal semantics” of the material – relates to the higher-level, “external” semantics of pervasive computing. A particular type of augmented material (for example a rigid plastic) can be used to form any number of classes of objects, each of which will exhibit different user affordances and behavioural relationships. This means that a material must know what it is externally as well as knowing its internal structure.

The external semantics is provided by describing the interaction rules by which the material should interact with other materials to which it is connected. This common use of rule bases leads to a single level of programming, built on top of an application-neutral sensing, communications and reasoning architecture. The external interactions of a material can use standard approaches such as web services and RDF (in its XML transfer format). This makes it easier to integrate artefacts using augmented materials into the wider context of pervasive computing.

4.4 Current State

We are exploring a number of possible programming approaches ranging from the pure logic programming outlined above to a more flexible approach based around domain-specific programming languages[8], building on the substrate of a compositional language technology based on Scheme[9]. The advantage of this approach is that we can leverage the benefits of type systems and language constructs while targeting micro-scale devices.

5 Related Work

One of the most relevant research activities on autonomous systems worldwide is the ”smart dust” project at the University of Berkeley. In this project, the goal is to develop a system of wireless sensor modules where each unit is the size of a mote of dust of around 2mm × 2.5mm – although each mote still needs external power and antenna, and so in practice is somewhat larger. This sets the design challenge for augmented materials, focusing on the design of miniaturised antennæ and power harvesting as well as power management and programmability.

The essential problem of communication within *ad hoc* networks has been the subject of vigorous research within the wireless sensor community. While a number of authors have studied localisation using infrastructurally-positioned beacons, recent work (for example [10, 11]) has addressed the problem of self-localisation of a group of nodes with sparse knowledge of their initial positions. We believe that this work may be extended to address augmented materials, with the crucial simplification that edge effects may provide additional localisation cues.

Software for pervasive systems ranges from the conventionally-structured desktop- and room-oriented approaches typified by the Context Toolkit[12] to the more chaotic approach of amorphous computing[6]. Although the latter remains largely conceptual, we believe that it provides good pointers to environments for augmented materials.

Some notable work has taken place under the European Union's "Disappearing Computer" initiative. The Extrovert Gadgets project integrated sensor networks into everyday objects. The GLOSS project investigated co-ordinating the global behaviour of pervasive systems. These (and other) projects have helped establish the broader landscape within which augmented materials must function.

6 Conclusion

We have presented a general approach to the problem of augmenting materials with embedded sensing and processing elements in a way that can be used to construct artefacts that combine physical and informational capabilities. This involves answering two distinct but related sets of questions:

- what are the hardware considerations in terms of location, communication, sensing and power involved in building a co-operative network of sensor elements?, and
- what is the appropriate programming model for applications on such a constrained platform?

Our tentative answers to these questions form the basis for an architecture for augmented materials in which a heterogeneous collection of low-power elements is co-ordinated by means of a hierarchical context model programmed in a highly declarative, whole-material style. We believe that this combination provides a good trade-off between the constraints encountered in building such augmented materials.

While our work is at an early stage, we have demonstrated the adequacy of the individual components and the way in which co-design can be used to leverage the best possible functionality from even limited individual units in a way that will scale to large materials. A fuller demonstration is currently underway to integrate hardware and software into a sample substrate to explore the ways in which augmented materials can complement and advance the capabilities of pervasive computing.

References

1. Bellis, S., Delaney, K., Barton, J., Razeeb, K., O'Flynn, B., Majeed, B.: Development of field-programmable modular wireless sensor network nodes for ambient systems. Computer Communications (2005) To appear.
2. Razeeb, K., Bellis, S., O'Flynn, B., Barton, J., Delaney, K., O'Mathuna, C.: A hybrid network of autonomous sensor nodes. In: Proceedings of the Second European Symposium on Ambient Intelligence. ACM International Conference Series (2004) 69–70
3. Dobson, S., Nixon, P.: More principled design of pervasive computing systems. In Bastide, R., Roth, J., eds.: Human computer interaction and interactive systems. Volume 3425 of Lecture Notes in Computer Science., Springer Verlag (2004)
4. Wang, X., Dong, J.S., Chin, C.Y., Hettiarachchi, S.R., Zhang, D.: Semantic Space: an infrastructure for smart spaces. IEEE Pervasive Computing **3** (2004) 32–39
5. Lassila, O., Swick, R.: Resource Description Framework model and syntax specification. Technical report, World Wide Web Consortium (1999)
6. Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, T., Nagpal, R., Rauch, E., Sussman, G.J., Weiss, R.: Amorphous computing. Communications of the ACM **43** (2000) 74–82
7. Montresor, A., Jelasity, M., Babaoglu, O.: Robust aggregation protocols for large-scale overlay networks. In: Proceedings of the International Conference on Dependable Systems and Networks. (2004)
8. Nisbet, A., Dobson, S.: A systems architecture for sensor networks based on hardware/software co-design. In Smirnov, M., ed.: Proceedings of the 1st IFIP Workshop on Autonomic Communications. Volume 3457 of LNCS., Springer Verlag (2005)
9. Dobson, S.: Creating programming languages for (and from) the internet. In: Workshop on Evolution and Reuse of Language Specifications for domain-specific languages, ECOOP'04. (2004)
10. Bulusu, N., Heidemann, J., Estrin, D., Tran, T.: Self-configuring localization systems: design and experimental evaluation. ACM Transactions on Embedded Computing Systems **3** (2004) 24–60
11. Wang, X., Silva, F., Heidemann, J.: Infrastructureless location-aware configuration for sensor networks. In: Workshop on Mobile Computing Systems and Applications, IEEE Press (2004) 174–183
12. Salber, D., Dey, A., Abowd, G.: The Context Toolkit: aiding the development of context-enabled applications. In: Proceedings of the ACM Conference on Computer-Human Interaction, CHI'99. (1999) 434–441

A Simulation Model for the Dynamic Allocation of Network Resources in a Competitive Wireless Scenario

Fernando Beltrán and Matthias Roggendorf

University of Auckland Business School,
7, Symonds Street, Auckland, New Zealand
{f.beltran, m.roggendorf}@auckland.ac.nz

Abstract. Next-generation wireless networks will enable the usage of different network technologies fully transparent to the user. Applications will be able to dynamically adapt to the conditions and technical constraints of the network. This vision requires a dynamic allocation of scarce network resources to different users. This paper presents simulation results from a model of admission control and dynamic resource allocation in wireless networks, in a two-provider, multiple-user scenario. The access allocation and connection procedure is implemented using an efficient (welfare maximizing) incentive mechanism for capacity allocation at both providers.

1 Introduction

The term Next-Generation Wireless Network (NGWN) describes a vision of a truly seamless, multi-technology, multi-provider network, serving mobile and fixed users with a variety of multi-media services. The major factors in achieving integration, flexibility and efficiency in NGWN are: the seamless integration of different access technologies, a high-performance physical layer, the adaptability of services and applications, and access to the core network, which must be flexible and adaptive [1]. Berezdivin et al. [1] state that the adaptation of application and services is the central element and is equated to the need that each element best uses its resources.

Application and service adaptation will pose many challenges to network operators. One of the prevailing questions is the dynamic allocation of scarce network resources in such a heterogeneous system. Wireless networks will introduce the possibility of having simultaneous access to competing networks, allowing users to seamlessly switch between network providers, even during ongoing service sessions. While in most mobile voice and data networks users are authorized by a subscriber identity module (SIM card), user authorisation in all IP-based networks can be designed much more flexible. Virtual mobile network providers, for example, purchase access capacities from different mobile networks and offer them to end customers. The aspect of multiple access has not yet been modeled in a dynamic allocation scheme.

The goal of this paper is to describe a model of access competition in a wireless multi-provider setting, while presenting its implementation in a simulated environment. Furthermore, the paper highlights the main conclusions arising from the results provided by simulation runs.

The simulation scenario is comprised of mobile agents – the users in the system – competing for access to two rival network providers. The mobile agents are allowed to adjust some resources to the agent's best use. Given the assumption of a competitive access framework, users' decisions on access will be bounded by their own resource-consumption levels and the contract conditions offered by the providers.

The rest of the paper is organised as follows: Section 2 briefly surveys the recent literature on the dynamic pricing of resource allocation in wireless networks. Section 3 introduces a two-provider multiple-user model in which users roam seeking connection to one of the two networks. The focus of the model is on the design of an incentive-compatible mechanism that assures the achievement of a measure of benefit once the allocation of a network resource such as bandwidth is granted. Section 4 presents the architecture and design of the simulation enviroment. Section 5 discusses the simulation results and provides an interpretation of the results derived. Section 6 draws a conclusion from the first results and depicts the next steps planned.

2 Dynamic Pricing for Resource Allocation in Wireless Networks

Dynamic resource allocation mechanisms have been applied in a variety of situations in wireless networks. Because the wireless transmission channel inherits additional complexity, such as varying channel quality and interference between networks and users, concepts from fixed networks cannot be transferred without careful consideration.

In order to organise the existing approaches to dynamic pricing in wireless networks we propose a simple layer model, distinguishing six abstraction levels of resource allocation (Figure 1). The horizontal layers representing the network model are mapped across the columns depicting the specifics of different wireless technologies up to the packet layer. The IP layer is seen as the first unifying layer, which provides full transparency to the above layers. On the lowest level of the layer model we find approaches dealing directly with the physical properties of the wireless channel. The emergence of Code-Division-Multiple-Access (CDMA) networks has pushed the need for sophisticated power management schemes that distribute the transmission power between active users in the network cell. Advanced mechanisms for allocating the power to different users can significantly increase the channel capacity allocation per user [2].

In [3], the authors design an allocation model for the uplink of a wireless mixed voice-and-data network in a two cell scenario. Voice users' demands are seen as inelastic while data users are "elastic users" who can assign a value to different data rates allocated to them. The optimisation problem is reduced to finding the maximal acceptance radius for voice users, depending on the signal attenuation, and the power allocation for data users.

A characteristic of wireless networks is the varying quality of the transmission channel. Many parameters, such as the structure of buildings or the speed of user movement influence the error rate and the transmission speed of the wireless connection. [4] study the use of pricing for downlink communication from a single wired network access point, focusing on the maximisation of the providers' revenue subject to the varying quality of the channel. In the extreme case, network providers would always provide all capacity to the users with the best channel quality to maximise throughput and, therefore, revenue. [4] introduces a first-price auction to allocate bandwidth among users, who are able to receive data, even with a bad channel, when they are willing to pay a higher price. This also means, that users may be charged different prices per time slot depending on their situation and location.

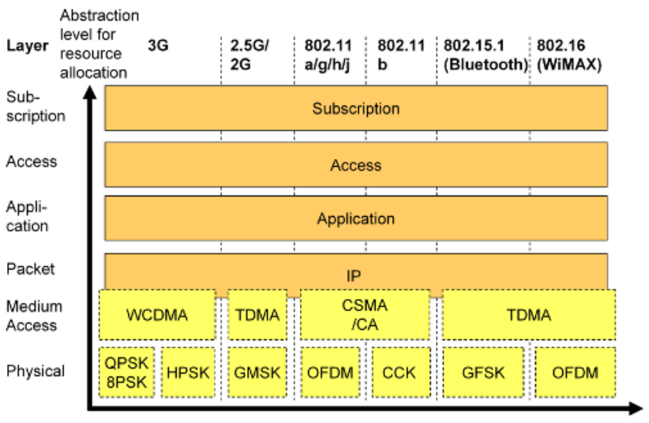


Fig. 1. The layer model to structure existing approaches for resource allocation and dynamic pricing in wireless networks

A different approach is chosen by [5]. An all-pay auction mechanism is proposed to distribute the available channel time. To illustrate the properties of the scheme they use a one cell, two user scenario with no interference between the users but different channel states which determine the throughput. For each time slot an auction is held. They show that the allocation leads to a Nash equilibrium which is no worse than 3/4 the maximum possible throughput when fairness constraints are not imposed and all slots are allocated to users with the best channel quality.

On the upper levels of the layer model only a few market-based allocation approaches can be identified (see e.g., [6] and [7]). These approaches focus at least in some aspects on user transparency and price predictability. While this aspect has already been discussed for broadband networks [8], comprehensive models for next-generation wireless networks are still missing.

3 A Model of Resource Allocation in a Wireless Multi-provider Setting

We introduce a model for the dynamic allocation of resources in a competitive wireless framework consisting of two wireless providers and a mixture of users. We assume all demand functions are quasi-linear. The allocation is dynamic in the sense that users can enter and leave the service areas at any time. Both providers use the Progressive Second-Price auction [9] as a mechanism to allocate bandwidth. The dynamics of connection and disconnection have an impact on the relative prices that providers charge for admission to and resource allocation in their networks.

3.1 Three Problems Arising from a Wireless Multi-provider Setting

The framework proposed here raises three problems. One is faced by any user who has to decide which provider to purchase capacity from. This decision comes after learning the results of a competitive bidding process that involves all users.

A second problem is the provider problem. Each of the two providers must realize that users may learn some information from their networks before making their decision about connection. Thus, a network making an announcement may expect users to accept the announced price or not.

In our model, users get a reply from each provider and then make a decision; providers receive bids and then communicate results to the users; a provider is also able to infer that he is losing a customer to the competing network when users who bid for access decide not to connect. Making sure users learn in a timely fashion such information is the challenge posed by the coordination problem.

3.2 The Progressive Second-Price Auction

Semret [9] proposes a game theoretic approach to the objective of a more efficient and fair utilization of shared resources; such an approach, called network games, results in mechanisms where intelligence and decision making is distributed. In a network the interacting agents acquire resources from the network on behalf of applications which need bandwidth and buffer space. Outcomes as efficient as those of a central controller may be collectively achieved if appropriate rules of interaction are introduced. Semret proposes that pricing can be resolved within the engineering of the network, overcoming the ex-post price structure generally imposed on most networks; his mechanism is called the Progressive Second-Price (PSP) auction.

The PSP auction is based on two aspects of mechanism design: realization and Nash implementation. Realization means the design of a message process (exchange of information between agents and the centre) that enables the achievement of a certain objective. Nash implementation means that allocation rules are designed with incentives, driving the players to an equilibrium where the desired allocation is achieved. This means that no agent can improve her situation by submitting a new bid without worsening the situation for another agent.

The PSP auction is an application of the Generalised Vickerey (GV) auction to allocate divisible objects (in this case, network resources) among bidders. PSP adopts this concept by letting all users submit information consisting of two values: the desired share of the total resource and the price they are willing to pay for it. The auction consists of players submitting bids (player i declares his desired shared q_i of the total resource and a price p_i he is willing to pay for it) and the auctioneer allocating a share a_i of the resource to player i at the cost c_i. The PSP auction allocation rule assigns player i a bandwidth a_i which is equal to the minimum value between his capacity bid, q_i, and the remaining capacity after all those capacity bids, q_k, whose prices beat (or are equal to) $i's$ bid ($p_k \geq p_i$) are subtracted from the total capacity to be allocated Q. In other words, the allocation rule is:

$$a_i(s) = q_i \wedge \left[Q - \sum_{p_k \geq p_i, k \neq i} q_k \right]$$

and s represents the set of bids by i, denoted as s_i and by the rest of the players, denoted as s_{-i}. The payment by any agent i is a weighted average of the (unit) prices offered by the other agents; each weight is the incremental capacity from including j in the auction. The pricing rule can be written as:

$$c_i(s) = \sum_{j \neq i} p_j \left[a_j(s_{-i}) - a_j(s_i; s_{-i}) \right]$$

The auction gets active when a new user attempts to join the network or another active user leaves. Both such events trigger the search for a new equilibrium and prompt users to start the submission of new bids. In order to guarantee the convergence of the algorithm a bidding fee ϵ has been introduced to let bidders change their bids only when the gain in net benefit is large enough. This is expressed in [9] as a modified concept of equilibrium known as ϵ-Nash equilibrium. From a technical perspective, the algorithm produces a minimum of signalling overhead since only two values have to be submitted.

4 Description of the Simulation Architecture

The simulation environment has been developed in the JADE framework. JADE provides a middleware concept to set up multiple, independently acting agents. Each market participant, user or provider, has been modelled as an agent with a specific behaviour profile defined at the beginning of the simulation. The JADE communication protocol enables communication between all agents as well as provides a discovery service to dynamically identify other market participants offering or requesting services.

A GUI-enabled agent processes all user input about agent types and agent profiles at auction startup and provides a status monitor at auction runtime to display results and performance parameters. Once the simulation has been started, two intermediary management agents manage the creation of agents and document simulation results in a central SQL database. At the beginning of the simulation all agents marked as resource providers (auctioneers), which offer network resources on the market, disclose their service parameters in a directory service. Bidding agents can then identify suitable network services from this directory and can subscribe at the respective auctioneers to participate in the bidding process.

4.1 The Bidding Process

At the start of the simulation bidders are able to discover all auctioneer agents in their range. As soon as they get registered at the auction they receive feedback about ongoing auctions and the bids submitted by other participants. Bidder agents can submit a bid to an arbitrary auctioneer agent. After submitting one or more bids a bidder agent "sleeps" for a pre-defined time period before it again observes the auction results. Whenever the bidder receives a new auction result from one of the auctioneers, a process is started to analyse the result and to calculate a new truthful reply for the next auction round. The truthful reply is based on the information received from the last auction round, namely, the bids from all other agents participating in the auction. In a first step, the bidder determines the quantity, at which it is able to compete against other bidders at its given demand profile. In a second step, the unit price is determined, which is the bidders marginal value.

4.2 The Auctioning Process

Since auctioneers act independently and do not cooperate with other network providers the auctioning process is conducted in a similar way as for the single provider scenario. As soon as an auctioneer agent receives a new bid it conducts a PSP auction and sends the results back to all participating bidder agents. The information sent back contains the individual resource quantity allocated to the bidder together with the total costs. Additionally, each bidder receives information of the bids submitted by each bidder.

4.3 Description of the Simulation Setup

The basic auction setup consists of five bidders requesting network resources from two auctioneers (table 1). Bidders' demands do not depend on the type of service requested. Table 1 shows the maximum quantity and maximum (unit) price for each user demand function. All users are in range of two available

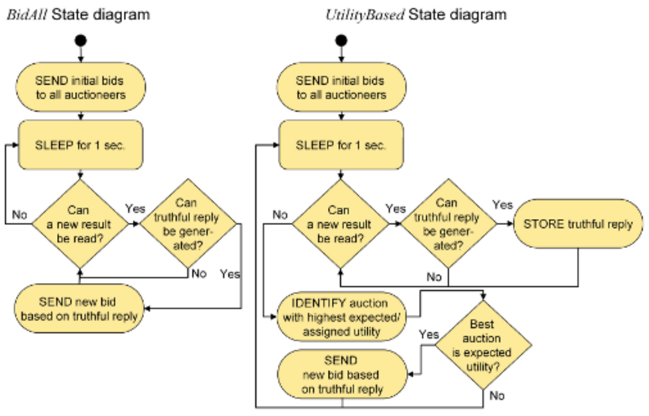
Table 1. Parameters for the first simulation experiment

Name	User1	User2	User3	User4	User5
Maximum quantity	90	85	80	70	65
Maximum Unit Price	10	12	15	20	22

wireless networks represented by two independently acting auctioneers. In the simulation the overall network resource Q is kept stable at $Q = 100$ but the distribution between the two auctioneers is varied from $Q_1 = 50, Q_2 = 50$ and $Q_1 = 100, Q_2 = 0$. It is assumed that there is no interaction or cooperation between the network providers. Network providers act independently and do not communicate either to maximize overall revenue or to provide shared capacity. The order of bids submitted in each round does not play any role in the final allocation result but can determine the number of auction rounds up to the final equilibrium at both auctions.[1]

4.4 The Simulation Sets

Two different sets of simulation runs were conducted, each with different bidding strategies applied. The first set of experiments assumes that users bid with their demand at both auctions without any coordination between the outcomes (called *BidAll*). It is also assumed that bidder agents can consume resources from several providers at the same time. When the two auctions have reached equilibrium users stay with the resulting quantities even if they have won resources from both auctions. This inhibits the risk of overbidding because, in equilibrium, resources were assigned from both auctioneers and bidders did not "coordinate" their demand accordingly. In extreme cases, this behaviour could lead to the winner's curse. However, with high congestion as in this case and the particular bidder profiles chosen, the winner's curse is unlikely to happen.



BidAll State diagram
SEND initial bids to all auctioneers
SLEEP for 1 sec.
Can a new result be read?
No
Yes
Can truthful reply be generated?
No
Yes
SEND new bid based on truthful reply
UtilityBased State diagram
SEND initial bids to all auctioneers
SLEEP for 1 sec.
Can a new result be read?
No
Yes
Can truthful reply be generated?
Yes
No
STORE truthful reply
IDENTIFY auction with highest expected/assigned utility
Best auction is expected utility?
Yes
No
SEND new bid based on truthful reply

Fig. 2. The simplified state representation of the bidder agent

The second set assumes that bidders coordinate their behaviour between both auctions (called *UtilityBased*). Instead of sending out a truthful reply only

[1] Since we conduct our simulations in an agent-based environment, certain external conditions can influence the timing of bids submitted in each simulation run.

based on the signals received from this particular auction, bidders compare the net benefit of both auctions. The bid with the highest net benefit is selected and sent to the respective auctioneer. If no new truthful reply can be generated but the bidder decides to stay with the existing bid from one of the last rounds, no message is sent to any of the auctioneers. Sending a new bid only to the one auction with the highest net benefit does not mean that the bidder resigns from other auctions since, once submitted, bids stay valid for future auction rounds. Even if no new bid is submitted the bidder is always informed about new results to be able to generate a new truthful reply.

Figure 2 depicts the bidding strategies of both simulation sets as a simplified state diagram.

For both scenarios 50 simulation runs were conducted. In each run the distribution of resources between the two providers was changed from $Q_1 = 51, Q_2 = 49$ to $Q_1 = 100, Q_2 = 0$. When the equilibrium at both auctions was reached, the values for revenue and consumer surplus were recorded by each bidding agent. Social welfare is defined as the sum of the revenue and the consumer surplus for each simulation run and is measured in a monetary unit.

5 Discussion of the Simulation Results

A linear optimisation model was used to calculate the maximum revenue at 1,456.2. Figure 3 displays the allocation of benefits between providers and consumers as the share of resources between the two providers changes from 50 - 50 to 100 - 0.

A prominent result is that the total welfare generated by different combinations of proportions in which providers supply the access market comes closer to the maximum as one seller's share gets relatively larger than the other's. There is some loss in efficiency when the market is equally supplied in comparison to the (monopolistic) one-provider situation. In the worst case such loss does not exceed 5% of the maximum revenue.

A second observation is that large differences in the providers' revenues can be observed between the two sets of simulation results. (See Figure 4) Joint revenues in both simulations decrease and get closer as the relative difference in shares of the supplied resource gets larger. As a consequence consumers' surplus exhibits a tendecy to improve, with some exceptional cases when UtilityBased strategy is used by the bidders.

Since agents seem to bid more carefully when using the *UtilityBased* strategy, in the sense that they only submit a new bid when it provides a higher utility than the current bid does, we would expect such strategy to improve consumers' surplus over the *BidAll* strategy. However, when providers equally supply the access market, *UtilityBased* yields more revenue to them than *BidAll* does, and as the difference in the relative shares of supply become larger revenue differences become smaller.

The latter can be explained by noticing that since under *UtilityBased* an agent switches between auctions, bids submitted to an auction in earlier rounds

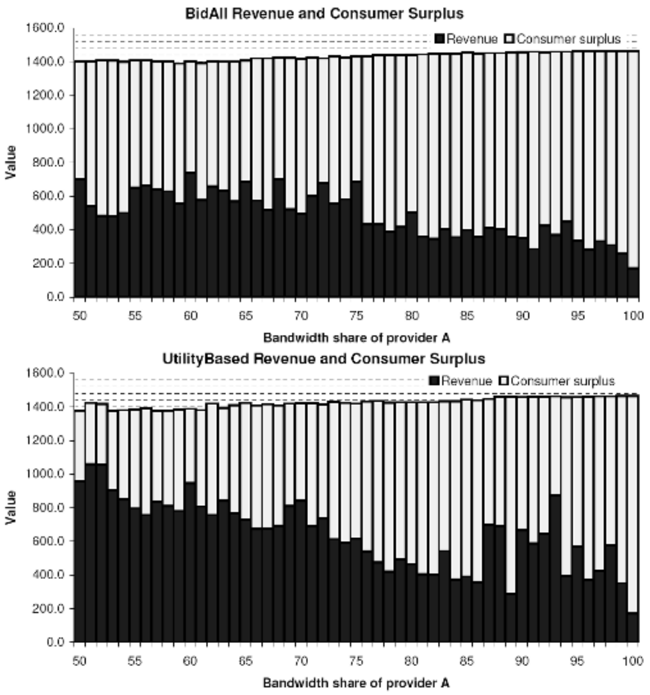


Fig. 3. Results of simulation set 1 and set 2 (consumer surplus, supplier revenue, and overall social welfare) for the *BidAll* bidding strategy and *UtilityBased* bidding strategy

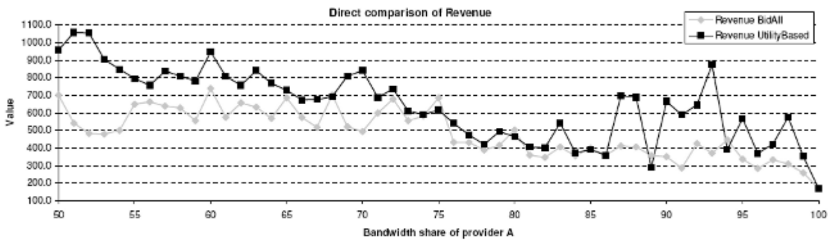


Fig. 4. Comparison between supplier revenue achieved with experiments of set 1 and set 2

stay active and influence the price level in the auction. The pricing rule of the PSP auction works on the basis of an exclusion-compensation principle, which determines prices while attempting to cover the "social opportunity cost" created by a bidder's presence. Because a bid remains active in the auction that did not report the highest utility to the bidder, the compensation still takes place at such auction. Consequently, as the auction rounds progress, prices do not decrease as fast as in the single-auctioneer PSP auction.

6 Conclusion and Next Steps

Our framework presents a proposal as to how to organise dynamic resource allocation in competing wireless networks based on an existing, social welfare maximising mechanism. Many challenges are ahead to transform this first approach into a comprehensive framework for dynamic resource allocation in competing wireless networks. An immediate next step is the improvement of the bidding mechanism using multibidding [10]. This has been shown to eliminate the convergence-time problem in a single PSP auction.

An unresolved issue is still how to model the bidding behaviour of agents bidding accross several auctions. Some promising proposals have been made for single objects in [11]. We intend to transform the framework of the problem space of this research to one in which bidding for divisible objects at different auctions is possible, avoiding overbidding (see also [12] for details).

As described, the PSP auction is efficient as it maximises the total social benefit. A rich source of problems opens if revenue maximisation, instead of social welfare maximisation is considered. Exploring the research problem from this perspective will also attract the attention of wireless network operators to explore future pricing models of next-generation wireless networks.

References

1. Berezdivin, R., Breinig, R., Topp, R.: Next-generation wireless communications concepts and technologies. IEEE Communications Magazine **40** (2002) 108–16
2. Saraydar, C., Mandayam, N., Goodman, D.: Efficient power control via pricing in wireless data networks. IEEE Transactions on Communications **50** (2002) 291 – 303
3. Zhou, C., Honig, M., Jordan, S., Berry, R.: Utility-based resource allocation for wireless networks with mixed voice and data services. In: IEEE International Conference on Computer Communications and Networks, Miami, Florida, USA. (2002) 485–488
4. Marbach, P., Berry, R.: Downlink resource allocation and pricing for wireless networks. In: IEEE INFOCOM 2002. (2002)
5. Sun, J., Modiano, E., Zheng, L.: A novel auction algorithm for fair allocation of a wireless fading channel. In: Conference on Information Science and System, Princeton, NJ, March, 2004. (2004)
6. Yaipairoj, S., Harmantzis, F.: Dynamic pricing with "alternatives" for mobile networks. In: Proc. IEEE Wireless Comm. Networking Conf. (WCNC). (March 2004) 457–470
7. Lin, H., Chatterjee, M., Das, S.K., Basu, K.: Arc: An integrated admission and rate control framework for competitive wireless cdma data networks using non-cooperative games,. IEEE Transactions on Mobile Computing **4** (2005) 243–258
8. Reichl, P., Stiller, B., Ziegler, T.: Charging multi-dimensional QoS with the Cumulus Pricing Scheme. In: Proceedings of ITCom 2001, Denver, Colorado, August 2001. (2001)

9. Semret, N.: Market mechanisms for network resource sharing. PhD thesis, Columbia University, Center for Telecommunications Research (1999)
10. Maille, P., Tuffin, B.: Multi-bid auctions for bandwidth allocation in communication networks. In: 23rd IEEE Infocom Conference, Hong-Kong, China. (2004)
11. Zeng, D., Cox, J., Dror, M.: Coordination of purchasing and bidding activities across markets. In: Proceedings of the 37th Hawaii International Conference on System Sciences. (2004)
12. Beltran, F., Roggendorf, M.: Defining optimal bidding strategies in competitive auctions of divisible objects. (Unpublished working paper, available on http://dream.com.auckland.ac.nz)

An Adaptive Call Admission Control to Support Mobility in Next Generation Wireless Network

Joo-Hwan Seo, Jung Seok Lee, and Ki-Jun Han

Department of Computer Engineering, Kyungpook National University, Daegu, Korea
horsedan@hanmail.net, leejs@netopia.knu.ac.kr, kjhan@bh.knu.ac.kr

Abstract. In this paper, we suggest that the next generation wireless network is the integrated cellular and ad hoc network. We propose a call admission control scheme to support burst handoff call using the guard channel and relay function in next generation wireless network. We evaluate our scheme in terms of new call and handoff call blocking probability and channel utilization via simulation study. Simulation results show that our scheme offers better performance than the conventional schemes. In addition, it is also shown that our scheme may efficiently balance traffic load when the network is congested.

1 Introduction

Most of the current wireless networks operate in the cellular-based infrastructure mode, in which all traffic goes through a base station or access point. Recently, significant progress has been made in a class of peer-to-peer wireless networks, called ad hoc or multi-hop wireless networks. Naturally, interoperability among heterogeneous networks is required for connecting ad hoc networks and wireless LANs to Internet through cellular network. There are many technical issues to be considered for integration of cellular and ad hoc networks, including relay, mobility management, balancing congestion traffic loads, Call Admission Control (CAC), and channel allocation for burst handoff call.

Much research effort was focused on relay technologies and balancing traffic loads, but there has been little study on problem to guarantee QoS of burst handoff call in a congested cell. In fact, such interoperability creates even heavier traffic in cellular system and burst traffic due to group mobility of ad hoc network, but the conventional call admission control schemes are not feasible to support the QoS of handoff call traffic and balance congestion.

In this paper, we propose an efficient call admission control scheme to guarantee QoS of a burst handoff call and balance traffic loads and in a congested cell. Our scheme keeps the handoff call blocking probability below some threshold by using relay functions and adjusting the amount of guard channel based on traffic load conditions. We evaluate our scheme through computer simulation in terms of new call/handoff call blocking ratio, and channel utilization. Simulation results indicate that our scheme reduces handoff call blocking probability than the conventional ones.

2 Network Architecture Model

First, we briefly explain the basic operation and architecture of the *iCARB (integrated network of Cellular with Relay technologies and Ad Hoc with Burst traffic).* In *iCAR(integrated Cellular System and Ad hoc Relay technologies)*, a number of ARS's (Ad hoc Relay Stations) are placed at strategic locations, and the ARS's can relay traffic between mobile hosts and base stations. The idea of "relay" was first introduced by Qiao et al. in *iCAR* [1-4]. Communications between a mobile host and a base station is done via the C-interface while those among mobile hosts are via the R-interface [1-4]. When traffic congestion happens in a cell, a mobile host that is initiating a new call cannot use its C interface to send its traffic to its base station. In this case, *iCAR* lets the mobile host use its R-interface to communicate with a nearby ARS, and the ARS can relay the signal through other base stations that still have available channels [1-4].

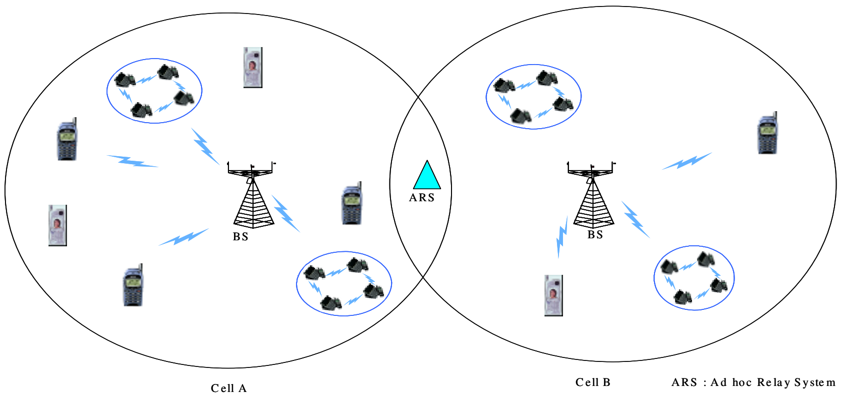


Fig. 1. Network Architecture Model

In our scheme, a handoff call is classified into two types: ordinary handoff call and burst handoff call. An ordinary handoff call occurs when one mobile node moves from one cell to another cell. A burst handoff call occurs when two or more mobile nodes with several ongoing calls move concurrently from one cell to another cell. A relay call is established when the current cell cannot allocate channel to a mobile node, and request its neighbor cell to service the mobile node. A relay call is also classified into two types: a new relay call and a handoff relay call. A new relay call occurs when a mobile node initiates a call in the current cell. When a handoff call is reversely relayed because there is no available channel in the congested cell, it is called a relay handoff call. A relay handoff call occurs when a mobile node processing an ongoing call relays the call through an ARS from the current congested cell to a neighboring non-congested cell.

In our scheme, we assume that a mobile node generates a new call, a handoff call or a burst handoff call in a heavily congested cell. In a congested cell, the mobile node searches an ARS for a relay route leading to a neighbor BS with low congestion control. Fig. 1 describes network architecture model used in this paper.

3 The Proposed Call Admission Control Scheme

Our scheme keeps the handoff call blocking probability close to its maximum allowable value while not deteriorating the new call blocking significantly. For this, our scheme determines the optimal number of guard channels depending on the current estimate of the instantaneous handoff call blocking probability.

In the conventional wireless network, when a mobile node generates a new call but the current cell does not have available channel(s) for it, the new call is blocked. However, the call does not have to be blocked in *iCARB*. In our scheme, a new call in the congested cell can switch over to the R interface to communicate with an ARS in a neighbor cell, possibly through other ARS's in the cell. This strategy establishes a relaying route between a mobile node and the BS in a nearby non-congested cell. If the neighbor cell has available channel(s), the new call is assigned. Otherwise, the relay call is blocked.

In the conventional wireless network, when a mobile node with ongoing calls moves to another cell, we check if there are any idle normal channel(s) available. If any, the handoff call is accepted and assigned the idle channel. Otherwise, the handoff call is blocked. Additionally, when the burst handoff call occurs because burst traffic moves concurrently from the current cell to another cell, we check if there are any idle normal channel(s). If there are as many as the number of the burst handoff calls, the all burst handoff calls are accepted and assigned the idle channels. If there are fewer available channel(s) than the number of the burst handoff call, the burst handoff call is partially accepted and assigned as many as available channel(s). The remainder handoff calls are blocked.

However, in our scheme, the remainder handoff call(s) try to switch over to R interface to communicate with an ARS in the neighboring cell, possibly through other ARS's in the cell. If the neighbor cell has available channel(s), the relay handoff call is accepted and assigned a channel. This call admission control procedure is illustrated in Fig. 2. In the proposed procedure, the amount of the guard channel is adjusted depending on the current network traffic condition when the handoff call blocking probability is estimated periodically.

Our scheme dynamically determines the number of guard channel(s) to keep the handoff call blocking probability close to the threshold value. Furthermore, our scheme tries to efficiently balance traffic load by guaranteeing the QoS of burst handoff call from the congested cell to the neighboring non-congested cell. Here, we define some parematers to formally explain our scheme such as table 1.

P_B is the handoff call blocking probability at time t and presented such as (1).

$$P_B = \frac{the\ total\ numberof\ blockedhandoffs}{the\ total\ numberof\ handoffs} \qquad (1)$$

Fig. 2 describes procedure of our scheme. Our scheme only selects the optimal number of guard channels according to (2) and (3) if the estimated handoff call blocking probability is blocked under the condition $P_B > \beta$. In addition, the optimal number of guard channels is determined according to (2) and (3) if the estimated handoff call blocking probability is blocked under the condition $P_B < \alpha$. A similar scheme has been applied in [14] for efficient bandwidth reservation.

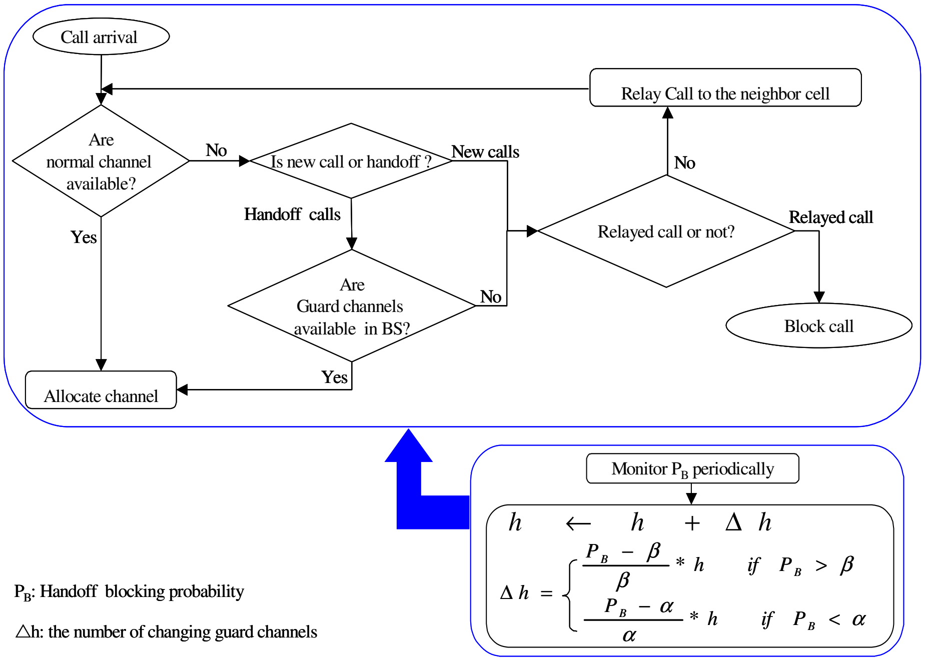

Call arrival
Relay Call to the neighbor cell
Are normal channel available?
No
Is new call or handoff ?
New calls
No
Handoff calls
Relayed call or not?
Relayed call
Yes
Are Guard channels available in BS?
No
Block call
Yes
Allocate channel
Monitor P_B periodically
$h \leftarrow h + \Delta h$
$\Delta h = \begin{cases} \frac{P_B - \beta}{\beta} * h & \text{if } P_B > \beta \\ \frac{P_B - \alpha}{\alpha} * h & \text{if } P_B < \alpha \end{cases}$
P_B: Handoff blocking probability
$\triangle$h: the number of changing guard channels

Fig. 2. Call admission control procedure of our scheme

Table 1. Parameters of proposed scheme

Parameters	Means
P_B	the handoff call blocking probability at time t
T_H	the maximum allowable handoff call blocking probability
$\lambda_h(t)$	the handoff call rate at time t
$\lambda_n(t)$	the new call arrival rate at time t
h(t)	the number of guard channels at time t
α_{up} and α_{down}	variables to adjust the guard channel to satisfy T_H
α	$T_H * \alpha_{down}$
β	$T_H * \alpha_{up}$

$$h \leftarrow h + \Delta h \tag{2}$$

$$\Delta h = \begin{cases} \dfrac{P_B - \beta}{\beta} * h & \text{if } P_B > \beta \\ \dfrac{P_B - \alpha}{\alpha} * h & \text{if } P_B < \alpha \end{cases} \tag{3}$$

The increment and decrement of the number of guard channels are both performed as soon as the threshold conditions for the blocking probabilities are satisfied via (2) and (3). Our scheme can automatically search the optimal number of guard channels to be reserved at a base station, and it can also be adapted to reflect traffic conditions such as changes in the call arrival rate.

We note that the proposed mechanism will only increase the number of guard channels as many as $\triangle$h when a handoff calls is blocked under the condition than $P_B > \beta$ and we only decrease the number of guard channels as many as $\triangle$h after handoff call blocking probability under the condition that $P_B < \alpha$.

Our scheme has the following important features. It periodically adjusts the number of the optimal guard channel depending on the blocking probability of handoff call. In addition, it tries to make sure that the handoff call blocking probability is below T_H by choosing $\alpha_{up} < 1$ and it tries to decrease the new call blocking probability by decrementing h(t) when it is observed to be more than needed.

The estimation period must be long enough to have a meaningful evaluation. If the estimation period too short, the system will change too often and certain measurements may not be accurate. On the other hand, if the estimation period is long, the system may not respond fast enough. In this paper, the estimation period is given 30 minutes.

4 Simulation and Results

We assume that there is no channel shortage along any relaying routes, and there is an ARS placed at each shared border of two cells, and its coverage is limited so that there is no overlap between any two ARS's. In other words, a mobile node will not be covered by more than one ARS. The ARS coverage is denoted by $0 < p < 1$. We analyze the performance of our scheme in *iCARB* with limited ARS coverage p. We assume that all normal free channels are fully available to handoff/new calls from the sources and all guard free channels are only available to handoff calls. The performance criteria include steady-state ratio of blocking of a new call, the steady-state blocking ratio of handoff call, and channel utilization, denoted by γ_{BN}, γ_{BH}, and ρ, respectively. These criteria are defined as follows.

$$\gamma_{BN} = \frac{the\ number\ of\ blocked\ new\ calls}{the\ total\ number\ of\ new\ calls} \tag{4}$$

$$\gamma_{BH} = \frac{the\ number\ of\ blocked\ handoffs}{the\ total\ number\ of\ handoffs} \tag{5}$$

$$\rho = time\ \ average\ \ of\ \ number\ of\ busy\ channels \tag{6}$$

In addition, traffic is evenly distributed in any given cell, hence, the probability that a call can be relayed to the neighboring cell (provided that it will not be blocked in the neighboring cell) is equal to the fraction ARS coverage p.

Table 2. Parameters of simulation model

Symbol	Mean	Value	Distribution
M	The number of total channel	30	
1/μ	Average call duration time	180 sec	Exponential
T_H	Maximum allowable value of handoff call blocking ratio	0.2	
λ_n	Mean arrival rate of new calls	3 ~ 13	Poisson
λ_h	Mean arrival rate of handoff calls	0.3* λ_n	Poisson
N_{bh}	Size of burst handoffs	1~10	Uniform
σ	Ratio of traffic load in two cells	1 : 0.5	
α	Lower threshold	0.7 * T_H	
β	Higher threshold	0.9 * T_H	
p	Success probability of relaying call	0.5	

We assume that the longest transmission range of a BS is 2Km and an ARS covers an area whose radius is 1Km. Important parameters used in simulation are listed in Table 2.

Simulation is performed in the model of two cells. In Fig. 3, we can clearly see that our scheme offers a much lower handoff call blocking ratio than the conventional dynamic one. Although the static guard channel scheme gives the lowest handoff call blocking ratio, it shows the worst behavior in channel utilization since the guard channel is fixed to 20 percentage of the total channel capacity.

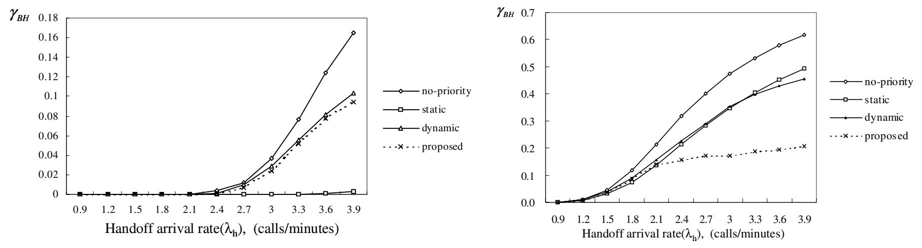


Fig. 3. Handoff call blocking ratio

Fig. 3 also shows that our scheme maintains handoff call blocking ratio below the maximum allowable value when burst handoff calls are generated. But, the conventional schemes shows a fluctuation around the maximum allowable value. Here, the size of burst handoff call is assumed to be uniformly distributed from 1 to 10.

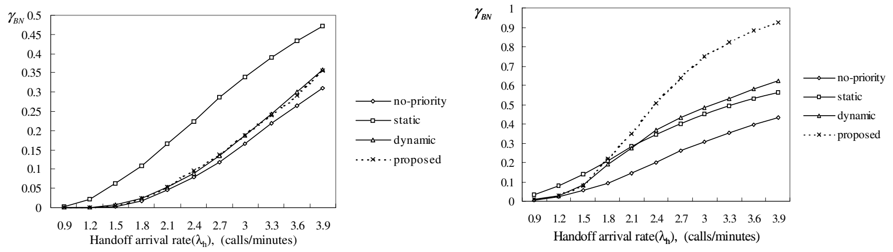


Fig. 4. Blocking ratio of new calls

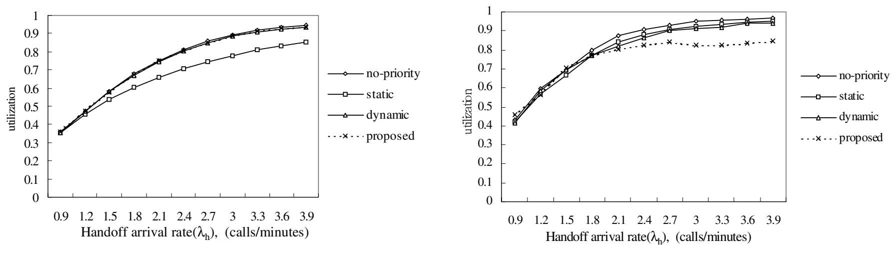


Fig. 5. Channel utilization

Fig. 4 indicates that with our scheme, the new call blocking ratio can be reduced but not by much. When the traffic load is low, our scheme offers a similar blocking ratio of new calls to the conventional ones. However, when traffic load is high, new call blocking ratio of our scheme is lower than the conventional ones.

Fig. 5 illustrates channel utilization of four schemes. This figure shows that our scheme gives a higher utilization than the static scheme at low traffic load. However, our scheme offers a little lower utilization than other schemes at high traffic load. This is because the guard channel increases to guarantee handoff call when the traffic load increases in our scheme.

5 Conclusions

In the integrated network of cellular and ad hoc, the conventional call admission control schemes do not efficiently guarantee the QoS of handoff call and are not feasible to support handoff call, in particular when the network is congested.

We propose an adaptive call admission control scheme to solve this problem. Our scheme can determine the optimal number of guard channels by using network traffic condition and handoff call threshold and relay characteristic in congestion cell. It can efficiently support burst call traffic and guarantees the QoS for burst handoff calls between cells.

We evaluate our scheme in terms of new call and handoff call blocking probability and channel utilization via simulation study. Simulation results show that our scheme offers better performance than the conventional schemes. In addition, it is also shown that our scheme may efficiently balance traffic load and guarantee the QoS of burst handoff call when the network is congested. However, the proposed scheme increases new call blocking ratio when traffic is congested.

References

1. Hongyi Wu; Chunming Qiao; De, S.; Tonguz, O., "Integrated cellular and ad hoc relaying systems: iCAR", *Selected Areas in Communications, IEEE Journal on* , Volume: 19 Issue: 10 , Oct 2001, Page(s): 2105 –2115.
2. Swades De; Tonguz, O.; Hongyi Wu; Chunming Qiao, "Integrated cellular and ad hoc relay (iCAR) systems: pushing the performance limits of conventional wireless networks", *HICSS. Proceedings of the 35th Annual Hawaii International Conference on.,* 2002, Page(s): 3931 -3938.
3. Wu, H.; Qiao, C.; Tonguz, O, "Performance analysis of iCAR (integrated cellular and ad-hoc relay system) ", *Communications, 2001. ICC 2001. IEEE International Conference on*, Vol. 2 , 2001,Page(s): 450 -455.
4. Jie (Jay) Zhou and Yang Richard Yang, "Parcels: Pervasive Ad-hoc Relaying for Cellular Systems", *In Proceedings of Med-Hoc-Net, Sardegna, Italy*, Sep. 2002.
5. X. Hong, M. Gerla, G. Pei, and C.-C. Chiang , "A Group Mobility Model for Ad Hoc Wireless Networks", *Proceedings of ACM/IEEE MSWiM'99*, Seattle, WA, Aug. 1999.
6. T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research", *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, Vol. 2, no. 5, 2002, Page(s): 483-502.
7. Guangyu, P.; Geria, M.; Xiaoyan Hong , "LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility", *Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on*, 2000, Page(s): 11 -18

8. Wang, K.H.; Baochun Li, "Group mobility and partition prediction in wireless ad-hoc networks", *Communications, 2002. ICC 2002. IEEE International Conference on* , Vol. 2 , 2002. Page(s): 1017 -1021.
9. Pei, G.; Gerla, M.; Hong, X.; Chiang, C.-C., "A wireless hierarchical routing protocol with group mobility", *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE* , 1999 , Page(s): 1538 -1542 vol.3
10. Veeraraghavan M; Karol M.J; Eng K.Y, "Mobility and Connection management in a Wireless ATM LAN," *IEEE Journal on Selected Areas in Communications*, Vol. 15 Issue:1, Jan. 1997, Page(s):50-68.
11. Yi Zhang and Derong Liu, "An Adaptive Scheme for Call Admission Control in Wireless Networks," *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, Vol. 6, 2000, Page(s):3628-3632.

Protecting Mobile Agent Loops

J. Cucurull, J. Ametller, J.A. Ortega-Ruiz, S. Robles, and J. Borrell

Department of Information and Communications Engineering,
Autonomous University of Barcelona,
08193 Bellaterra - Spain
`Jordi.Cucurull@uab.es`

Abstract. A mobile agent's itinerary describes the set of hosts visited during the agent's travel, and must be protected against malicious entities trying to access and/or modify it for their own benefit. Protection mechanisms must be put in place, but we should avoid imposing unnecessary limitations on the agent's ability to choose its itinerary in an as flexible as possible way. In this article, we extend previous work on itinerary protection protocols to include loops among the allowable protected itineraries for roaming agents. Our agents can thus traverse a whole new range of paths in a secure way.

Keywords: Mobile Agents, Security, Itinerary Protection, Loops.

1 Introduction

Mobile agent [14] systems are a well known paradigm in the distributed computing arena. Their ability to migrate partially completed computations together with their context, provide ample room for innovative applications in a variety of scenarios: high latency networks, ad-hoc and rapidly changing network topologies or high volume distributed data processing, being just a few examples of particularly fitting situations to the mobile agent paradigm.

The power of this new paradigm does not come for free, though. Mobile agents introduce new challenges and opportunities for mischief, which must be addressed before we can see real-world applications deployed using them. Security is probably the most trying aspect in mobile agent systems, and ensuring sound protection against external malicious parties in mobile agent environments is still an open problem. More concretely, a mobile agent owner must have the guarantee that the agent's code and data will be executed without external modifications. On the other hand, the execution environment must be protected from malicious agents.

Protection against malicious execution environments has proved an extremely tough problem [15]. In general, we must consider the scenario of multiple environments, or *hosts*, without any trust relation between them. As a matter of fact, they must be regarded as potentially competing entities, willing to interfere (by tampering with the agent's code and/or data) with each other to obtain any kind of advantage over competitors. We assume this entities won't try to destroy the other's agents. Many partial solutions to this problem have been proposed. Although probably none of them is without flaws or corner cases, the soundest are, non-surprisingly, based on cryptographic protocols (cf.refs in section 1.1).

These protocols use standard cryptographic mechanisms (such as public key encryption or digital signatures) to protect the agent code and data while it is traversing the hosts in its itinerary.

1.1 Itineraries and Loops

A mobile agent itinerary is defined as the set of platforms the agent visits to complete its execution. Note that itineraries may be non-deterministic: due to their autonomic behaviour, agents can discover new platforms during their travel, and choose their destination according to criteria or data not predictable at the launch point. In other words, agent itineraries are, in the general case, dynamic.

This lack of determinism makes cryptographic protection harder. Proposed solutions for itinerary protection usually require the existence of a key pair associated to each host the agent is visiting [8, 4, 2]. The host public keys are then used to cipher relevant agent data at its origin, making it available only to its intended target. Obviously, this strategy will only work if we know beforehand all the platforms the agent will visit, and, therefore, dynamic itineraries are greatly impeded.

Newer proposals [11, 6, 7] provide greater flexibility by allowing multiple itineraries through a pre-defined set of hosts. The latter is still fixed, but multiple traversal strategies are allowed.

An alternative protection strategy relies on the use of Trusted Third Parties (TTPs). Such proposals [3, 12] provide higher security at the expense of convenience: agents are required to visit said TTPs during their travel, complicating the system and reducing its efficiency and scalability.

At any rate, none of the mentioned solutions permits the existence of loops in the agent's itinerary. That is, agents cannot traverse iteratively a closed itinerary over a finite set of hosts. There are many situations where such routes would be desirable, particularly when the agent is negotiating with competing platforms offering a given service or product. In such an scenario, the hosts will be revisited to give them a chance of modifying their original offers, according to the offers of their competitors to the agent, which will loop through the platforms until an acceptable offer is obtained.

Allowing loops, though, introduces new security risks. If an agent can visit a platform multiple times, distinguishing legal visits from replay attacks by dishonest entities becomes a problem. In replay attacks, the agent is re-sent by a malicious interceptor to a previously visited platform, in order to disclose reserved information or otherwise affect the agent's execution outcome to the attacker benefit [9, 10, 12].

This article presents an extension of the protection protocol described in [7] to allow itinerary loops which are not vulnerable to replay attacks. The following section sums up the protocol of [7]. Our extended protection protocol is fully described in section 3, and its security analized in section 4. We close this article with our conclusions and plans for future work.

2 Itinerary Protection Protocol

In this paper, we shall extend the scheme for agent itinerary protection described in [7]. That scheme is flexible enough to allow arbitrary combinations of sequences, alternatives and sets to specify the protected itineraries over a fixed list of hosts. This section provides a brief summary of the base protocol that we intend to extend, and the notation used in the rest of the paper.

To protect an itinerary, its representation as a Petri net is first constructed. This construction allows an efficient treatment of complex paths, and provides a clear-cut specification of the transitions to be protected.

The protection protocol is based on digital envelopes [5] with the structure:

$$\mathcal{D} = (P_j(r_i)|E_{r_i}(I))$$

where r_i is the (randomly generated) envelope's symmetric key and P_j denotes the destination host's public key. The owner of the corresponding private key can therefore obtain r_i to decipher the information I. The protected itinerary is built as a chain of digital envelopes, so that it can only be disclosed in a pre-defined order. Let us describe how the protection protocol it works.

Protection Protocol:

1 Initialization : Itineraries are represented as a Petri net with n nodes labelled h_i for $1 \leq i \leq n$. The agent's owner, $\mathcal{O}$, generates n random keys, r_i, and assigns one to each node. Each node represents a host to be visited. We denote $t_{i,j}$ the transition from node i to node j.

2 For each transition $h_i \Rightarrow h_j$, create $t_{i,j}$:

(a) Signature: The owner signs the address of h_i and h_j, r_j and a travel mark t (see Eq. 3 below), obtaining:

$$S = S_{\mathcal{O}}(h_i, h_j, t, r_j)$$

(b) Transition token: The owner ciphers S using h_j's public key; the transition token $t_{i,j}$ is then obtained as the concatenation of this value and h_j address:

$$t_{i,j} = (h_j, P_j(S_{\mathcal{O}}(h_i, h_j, t, r_j))) \tag{1}$$

3 Information ciphering: The agent's owner proceeds to encipher each node's private information, including his next transitions, using the corresponding symmetric key r_i. This information will consist of a method or task, m_i and, for alternate entries, a condition, $\mathcal{C}$ (see figure 2). If we denote by E_i the operation of enciphering with key r_i we can write the ciphered information e_i, representing an alternative, as:

$$e_i = E_{r_i}(m_i, \mathcal{C}, t_{i,j}, t_{i,k}). \tag{2}$$

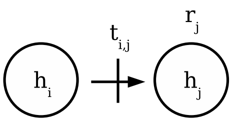


Fig . 1. Sequence type transition

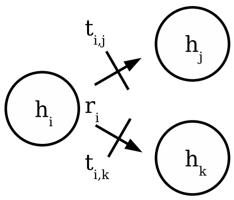


Fig . 2. Alternative type transition

4 End : Once the above steps have been performed for each node in the Petri net, we end up with a protected itinerary, which can be denoted as $I = (e_1, e_2, \cdots, e_n)$.

The travel mark t identifies the agent, and precludes replay and cut-and-paste attacks. It is defined as:

$$t = T \parallel H(\text{mobile agent's code}), \tag{3}$$

where T is a time stamp and H a cryptographic hash function.Without this mark, a dishonest host could change the agent's itinerary, using an old agent's one (belonging to the same owner). T guarantees that t is unique for each travel, while the agent's code hash binds the itinerary to its legitimate agent. Each visited host will store t only as long as the agent owner's signatures remain valid, so that the agent's validity is time-limited.

2.1 Properties of the Protection Protocol

The protection protocol just described can be applied to itineraries created by arbitrary combinations of sequential and alternate subpaths, and it will provide the following guarantees [7]:

P1 Integrity
P2 Confidentiality
P3 Forward privacy
P4 Data verifiability
P5 Originator's verifiability
P6 Strong identification
P7 Once entry

The P6 property, which guarantees that the agent is executed only once in each host, precludes loops in the itinerary. To allow them, we must provide a way of distinguishing legitimate re-executions (ensuing from the agent following a closed path during its travel) from malicious replays. The following sections show how this can be done without jeopardizing the protection scheme.

3 Protection Protocol for Loops

3.1 Protection Protocol Modifications

Let us consider the protection of itineraries containing loops, such as the one represented in figure 3. The existence of closed paths will obviously imply that the agent's code is executed more than once in one or more nodes of its itinerary. The challenge is to distinguish such legitimate re-executions from malicious ones issued by an attacker.

Let N denote the vector of maximum allowed agent executions at each itinerary node for any given time; i.e.

$$N = (seq_1, seq_2, \ldots, seq_n)$$

where seq_i is the maximum number of times the agent can visit host h_i during the rest of its travel. At the origin, this vector will be initialized with the maximum allowed visits to each node, and, each time the agent is executed at host h_i, its corresponding counter seq_i will be decremented. The key of our protocol is to include this vector in the information that travels with the agent. Rather than N, the agent will contain the execution counters in ciphered form, as the vector:

$$v_\omega = (E_{r_1}(seq_1)), E_{r_2}(seq_2), \ldots, E_{r_n}(seq_n)), \tag{4}$$

where we have made explicit this vector's dependency on ω, the step in the agent travel. That is, at the agent's origin we are at step $\omega = 0$, after visiting the first host in the itinerary, $\omega = 1$, and so forth until the travel ends for some value $\omega = \omega_f$.

E_{r_i} denotes the symmetric ciphering function using key r_i; as mentioned, only the agent's owner and h_i have access to the corresponding visits counter. The former will initialize the counter with the maximum allowable number of visits to the host for a given itinerary, while the latter will take care of checking and decrementing it after each visit.

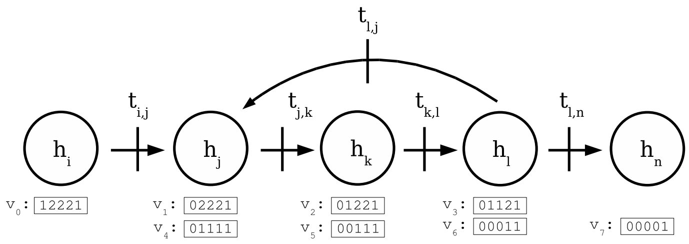


Fig. 3. One loop itinerary

The protection provided by the symmetric ciphering of seq_i can be further strengthened by adding to the counter random bits at fixed positions, which will be changed every time seq_i's value is recomputed in order to avoid predictability-based attacks.

In the example depicted in figure 3, if we wish the agent visits each internal loop nodes twice, we would have $seq_i = seq_n = 1$ and $seq_j = seq_k = seq_l = 2$, and, during the agent's itinerary, v_ω will take eight different values, i.e., ω will run from 0 to 7.

The first time an agent visits h_i the following steps will ensue:

– The host deciphers $E_{r_i}(seq_i)$ and the travel token t. Since this is the first agent's visit, no record containing t will be found in the host's database.
– A time to live, TTL, is assigned to the above values. This TTL will be greater or equal to the agent's expiration time, so that the agent cannot be re-executed after the stored counters have expired (and possible deleted from the system).
– The tuple

$$t_i = (t, seq_i, \mathrm{TTL}) \tag{5}$$

is stored.
– If $seq_i > 0$, the agent is executed, its counter decremented and v_ω updated accordingly.

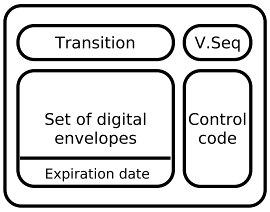


Fig. 4. Mobile agent components

On the other hand, every time the agent closes a loop and re-visits h_i the following process will take place:

- Extract from the vector v_ω carried by the agent its current execution counter, seq_i.
- If the agent has expired or $seq_i = 0$, the agent will be rejected.
- Retrieve the tuple t_i (cf. equation (5)) associated with this agent, using t as a search key.
- If agent's execution counter, seq_i, is not lesser than tuple's (t_i) associated counter, the agent is rejected.
- Otherwise, the agent is executed.
- t_i is updated substituting his counter for seq_i's agent counter.
- The agent's data is updated substracting one unit to seq_i in v_ω, and the agent migrates to its next destination.

A key ingredient of the above protocol is the vector v_ω carried by the agent. Figure 4 provides a schematic view of the main agent components. If v_ω, as given by equation (4), would be transported without further protection, a malicious third party could capture it and try to tamper with the system using substitution attacks (cf. section 4.1).

To avoid substitution attacks, the execution counter vector will be ciphered so that it can only be read by the next node in the agent's itinerary. In addition, the receiving node should be able to check the legitimacy of the agent's sender (i.e. the previous host in the itinerary, which was the last v_ω modifier). These goals can be met by storing in the agent, instead of v_ω, the following quantity:

$$p_i = P_j(S_i(v_\omega, H(t_{i,j}))), \tag{6}$$

where i is the sending host, j the receiver and $t_{i,j}$ the transition given by equation (1). Since the vector is ciphered using the receiver node's public key (this ciphering being denoted by the function $P_j()$ above), only the receiver can access it. The digital signature S_i provides verifiability of p_i's creator identity. Finally, the hash function H binds p_i to the corresponding transition.

This completes our new protocol definition. Note that the creation of p_i should be fully automated, avoiding specially access to the private keys of the visited platforms. To that end, the cryptographic services scheme proposed in [1] can be used.

3.2 Loop Implementation

Our new protocol can be implemented, in systems using itinerary protection mechanisms of the kind described in section 2, in a manner analogous to that of alternatives.

For instance, to implement the loop depicted in figure 3, we just need two additional transitions: one from the final host to the initial host ($t_{l,j}$) and one between the final and next hosts ($t_{l,n}$), which exits the loop. Both transitions will be signed by the agent's owner and ciphered for the corresponding destination node (as described in [7]):

$$t_{l,j} = (h_j, P_j(S_O(h_l, h_j, t, r_j))), \tag{7}$$

$$t_{l,n} = (h_n, P_n(S_O(h_l, h_n, t, r_n))), \tag{8}$$

In addition, these transitions will be contained in the digital envelope assigned to the last node in the loop, with the same format used for an alternative, namely:

$$e_l = E_{r_l}(m_l, \mathcal{C}, t_{l,j}, t_{l,n}), \tag{9}$$

where $\mathcal{C}$ denotes the guard condition to be fulfilled for the transition to the initial node to take place.

Finally, the execution counters will be initialized at origin with the corresponding values for the maximum number of allowed executions in each host.

4 Security Assessment

This section examines our proposed protocol resilience against external attacks.

4.1 External Itinerary Replay

In replay attacks, as depicted in figure 5, a malicious external entity capture roaming agents and try to modify and execute them at hosts on times different to the intended ones. Agents can be captured by network sniffing or collusion with a host in the agent's itinerary.

The use of a Public Key Infrastructure (PKI) and the associated encryption and verification services, precludes this kind of attacks in our protocol. These techniques are used also in the original itinerary protection protocols, so that both the static and dynamic (cf.Figure 4) parts of our agents are protected. As mentioned, the use of public

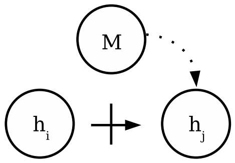


Fig. 5. External Reply Attack

key encryption and digital signatures to protect this data avoids any impersonation risks (a third party trying to play the part of a legitimate host).

Substitution attacks will also be detected. In this case, the attacker would try to send her own agent to resume the captured agent's itinerary. A new execution counter vector could be constructed with the correct structure, but it would nevertheless be detected as bogus when trying to get the agent counters, because symmetric keys will be different.

4.2 Internal Itinerary Replay

In this scenario, the attack is conducted by a dishonest host in the agent's itinerary, which tries to re-execute the agent in another node to the host's advantage (e.g., to make it buy more items than planned at the host's electronic shop). As shown in figures 6 and 7 this attack can be attempted either by an isolated host or in collusion with other hosts in the itinerary.

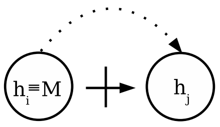


Fig. 6. Single host internal replay attack

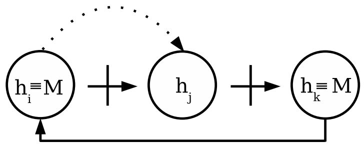


Fig. 7. Internal replay attack with collusion

Single host. It is easily seen than this attack has no chance of success. Despite having access to v_ω, the dishonest host cannot alter the execution counter of any other host in the itinerary, for it has no access to the symmetric key of the attacked host.

Colluding hosts. In this attack, two dishonest nodes collude to cause unauthorized agent executions on a third honest host (cf. figure 7). When the mobile agent reaches the second dishonest host, this host sends v_ω to the colluding platform (both of them will have access to its contents, via their private keys). The receiver will inject this new vector in a copy of the agent (which had visited the platform in a previous itinerary step), appropriately signed and ciphered, and resend the agent to the attacked host. The latter will have no way of detecting the fraud, and will re-execute the agent. In this way, unintended agent executions can be provoked and go undetected: our protocol cannot cope with collusion.

4.3 Internal Host Replay

In this attack, a dishonest host re-executes the agent as many times as it sees fit, disregarding the execution counter. As noted in the literature [12], this kind of attack is unavoidable, due to the fact that it is not observable from the outside.

4.4 Loop Iteration Replay

A final security breach arises when the host that decides when a loop needs to be iterated is dishonest. Such a host has a privileged role, and can provoke as many iterations of a closed loop as desired, unless the number of iterations is constant and can be fixed beforehand. In the latter case, the maximum number of executions set at the agent's origin will also be the total number of executions for any legitimate itinerary traversal, and the attack would be detected. In itineraries where the actual number of loop iterations is dynamically determined by the agent, the only solution is to ensure the critical host's honesty by external means.

5 Conclusions

Current itinerary protection mechanisms make no provision for closed loops in the mobile agent path, precluding in this way a whole range of interesting, dynamic use cases. In other words, these mechanisms enforce single execution per host itineraries.

This paper provides a solution to this problem by means of a new protocol that allows a bounded number of iterations or agent reexecutions in one or more nodes of its itinerary. In this way, and without sacrificing a rigidly controlled itinerary structure, mobile agents gains the ability to dynamically modify its path by re-visiting some nodes of its itinerary.

We have shown that the new protocol protects the agent against all external attacks, and some internal dishonest host behaviours. Internal collusion attacks, and unauthorized re-executions within a single platform are still possible, and constitute open problems of our proposal. Some work in this area has been done by [13, 12].

Some of these shortcomings are addressed by other proposals using Trusted Third Parties. But, in our opinion, such solutions imply a considerable overhead when it comes to their implementation and deployment in real systems, which will hamper their adoption. In our protocol, we have opted for offering a compromise between security and efficiency, with an eye on deployment feasibility.

Acknowledgments

This work has been funded by the Spanish Ministry of Science and Technology (MCYT) though the project TIC2003-02041.

References

1. J. Ametller, S. Robles, and J. A. Ortega-Ruiz. Self-protected mobile agents. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 362–367, Washington, DC, USA, 2004. IEEE Computer Society.
2. A. Corradi, R. Montanari, and C. Stefanelli. Mobile agents protection in the internet environment. In *23rd Annual International Computer Software and Applications Conference*, 1999.
3. M. Giansiracusa, S. Russell, A. Clark, and V. Roth. Macro-level attention to mobile agent security: Introducing the mobile agent secure hub infrastructure concept. In *Information and Communications Security: 6th International Conference, ICICS 2004*, volume 3269 of *Lecture Notes in Computer Science*, pages 343–357. Springer Verlag, 2004.

4. Neeran M. Karnik and Anand R. Tripathi. Security in the Ajanta mobile agent system. *Software Practice and Experience*, 31(4):301–329, 2001.
5. R. Laboratories. PKCS 7: Cryptographic message syntax standard, 1993.
6. J. Mir and J. Borrell. Protecting general flexible itineraries of mobile agents. In *Proceedings of ICISC 2001*, LNCS. Springer Verlag, 2002.
7. J. Mir and J. Borrell. Protecting mobile agent itineraries. In *Mobile Agents for Telecommunication Applications (MATA)*, volume 2881 of *Lecture Notes in Computer Science*, pages 275–285. Springer Verlag, October 2003.
8. V. Roth. Empowering mobile software agents. In *Proc. 6th IEEE Mobile Agents Conference*, volume 2535 of *Lecture Notes in Computer Science*, pages 47–63. Spinger Verlag, 2002.
9. Volker Roth. On the robustness of some cryptographic protocols for mobile agent protection. In *Mobile Agents : 5th International Conference, MA 2001 Atlanta, GA, USA, December 2-4, 2001. Proceedings*, volume 2240 of *Lecture Notes in Computer Science*, 2001.
10. Volker Roth. Programming satan's agents. *Electr. Notes Theor. Comput. Sci.*, 63, 2001.
11. M. Straßer, K. Rothermel, and C. Maifer. Providing Reliable Agents for Electronic Commerce. In *Proceedings of the International IFIP/GI Working Conference*, volume 1402 of *Lecture Notes in Computer Science*, pages 241–253. Springer-Verlag, 1998.
12. Y. Tsipenyuk and B. Yee. Detecting external agent replay and state modification attacks. http://www-cse.ucsd.edu/˜ytsipeny/home/research/paper.pdf, February 2005.
13. D. Westhoff, M. Schneider, C. Unger, and F. Kaderali. Protecting a Mobile Agent's Route against Collusions. In *Selected Areas in Cryptography: 6th Annual International Workshop, SAC'99, Kingston, Ontario, Canada, August 1999. Proceedings*, volume 1758 of *Lecture Notes in Comptuer Science*, page 215. Springer-Verlag, Jan 2001.
14. James E. White. Telescript technology: Mobile agents. In Jeffrey Bradshaw, editor, *Software Agents*. AAAI Press/MIT Press, 1996.
15. Bennet S. Yee. A sanctuary for mobile agents. In *Secure Internet Programming*, pages 261–273, 1999.

ContextWare Support for Network and Service Composition and Self-adaptation

Roel Ocampo[1,2], Lawrence Cheng[1], Zhaohong Lai[1], and Alex Galis[1]

[1] Department of Electronic and Electrical Engineering,
University College London, Torrington Place,
London WC1E 7JE

[2] Department of Electrical and Electronics Engineering,
University of the Philippines,
Diliman, Quezon City, 1101 Philippines

{r.ocampo, l.cheng, z.lai, a.galis}@ee.ucl.ac.uk

Abstract. A crucial prerequisite for building context-aware networks is an integrated infrastructure for sensing, processing, managing and disseminating network context information to network entities and user-facing applications. This requirement is especially important in ambient networks, a class of networks that exploit the inherent heterogeneity seen in today's wireless networks in order to share diverse resources and services across different but cooperating networks. We discuss ContextWare, our architectural solution for making ambient networks context-aware, and present a programmable approach to its realization. We relate some of our experiences in building and validating a prototype and demonstrating its support for network and service composition, as well as self-adaptation, using a reference scenario.

1 Introduction

Today's wide range of portable electronic devices provide powerful functionalities and offer a variety of methods for interconnection and information sharing. However, as people go about their daily tasks, equipped with a collection of these devices, they often are only able to enjoy the functionalities provided separately by the individual parts, rather than the integrated whole, of this collection. We sometimes discover to our dismay that such devices often only interconnect with like devices or are able to roam in a limited fashion within dedicated, specialized wireless overlays. This impairs user mobility, as users often have to search for and remain within the range of compatible devices, access points or gateways to these dedicated overlay networks.

Such a situation hinders the ability of users to access services found within these islands of connectivity in a seamless and end-to-end fashion, thus preventing them from realizing their full productive potential. It is inconvenient or even difficult for users to deal with such heterogeneity; it simply demands too much of their attention. Our view is that networks should be self-managing in order to relieve users of the burden of manually configuring and interconnecting them, and that they should self-adapt to continuous changes in the network environment. Networks should have the ability to sense the presence of other networks in the environment, their capabilities, and the services they may offer. They should have the ability to cooperate in order to provide seamless end-to-end services to users.

In this paper, we provide a brief introduction to *ambient networks* (ANs), a novel class of networks that embrace and exploit the inherent heterogeneity seen in today's wireless networks in order to share diverse resources and services across different but cooperating networks. We then discuss our efforts in providing ambient networks the ability sense and adapt to the network environment as well as the activities and needs of users, i.e., making ANs *context-aware networks*, by designing an integrated infrastructure for context-awareness within ANs called *ContextWare.*

The rest of this paper is organized as follows: in the next section we outline the concepts and architectural features of ambient networks. Section 3 discusses the role of context in ANs and our design approach in making ANs context-aware. Section 4 describes the design and implementation of a prototype of this architecture, while Sect. 5 relates our experiences in validating this prototype. Finally, Sect. 6 concludes and describes our work ahead.

2 Ambient Networks

The Ambient Networks Project (http://www.ambient-networks.org) is an integrated project co-sponsored by the European Commission under the Information Society Technology (IST) priority within the 6th Framework Programme (FP6). The concept of ambient networks evolves partly from the vision of "Ambient Intelligence" (AmI) where people are surrounded by intelligent, intuitive interfaces that are embedded in all kinds of objects, working in an seamless, unobtrusive and invisible kind of way [1]. To realize this, one basic mechanism employed in ANs is the dynamic, instant composition of networks, where interconnectivity, access and services are combined across different but cooperating network domains without user involvement [2]. ANs based on federations of multiple, cooperative networks integrate and interwork the capabilities of these different technologies and networks and abstract the inherent diversity, presenting an end-to-end and seamless network to the user.

Ambient networks are built on three key design principles: open connectivity and open networking functions; an incremental deployment of functionality on existing networks; and self-composition and self-management [2]. In ANs, a common set of control functions are built over the existing functions of legacy networks to enable end-to-end connectivity and service provisioning. These control functions are collectively hosted within a common control plane called an *Ambient Control Space*, or ACS. The ACS is an integral design feature of ambient networks; an "ambient network" is the union of a (possibly legacy) connectivity network with an ACS [3].

In Fig. 1a, we have two cooperating ambient networks composing into a single ambient network. Each ACS has an Ambient Network Interface (ANI) that advertises the presence of control space functions to adjacent domains, and offers standardized means for these functions to interoperate. The Ambient Service Interface (ASI) presents a single, homogeneous view of the composed ambient network and allows external entities such as users to access its services and control functions.

Self-composition and self-management are essential features of ANs as it would be impractical and counter-productive for humans to continuously monitor and manually manage the processes of network composition and decomposition, as well as end-to-end service provisioning, within such highly heterogeneous, dynamic, mobile network environments. This design principle serves as one of the primary motivations for the use of context within ANs, as we begin to discuss in the next section.

3 Context in ANs

Context plays a crucial role in ambient networks, and one of the main functions within the ACS in Fig. 1a is that of *context provisioning.* The main role of context in ANs is seen as:

> The challenge in AN is to make the networks context-aware, where software and hardware sensors are used to capture context information. Entities that will produce and use context information are for example routers, devices, mobile nodes, moving and vehicular networks ... This view of context management differs from the traditional focus on user parameters regarding applications and services; rather, it takes a *network-centric* view [4].

The specific motivations for introducing context-aware functionality within ANs are: first, to enable automatic decision-making by the functions within the ACS, thereby reducing complexity and enhancing performance; and second, to enable ACS internal functions and externally-accessible user services (called user-facing services) to adapt their operation based on context, optimizing network service delivery and functionality. These would be achieved by creating an infrastructure that allows the network to collect and make use of both user- and network-related context information to direct operation and decision-making by network entities [4]. The functional elements within the infrastructure that manage context in ambient networks, mediating between context sources and sinks, are collectively called *ContextWare* [5]. In the next few sections, we discuss the design requirements that have to be satisfied in order to create context-aware ambient networks; we then discuss an architecture for ContextWare that satisfies these requirements.

3.1 Motivating Scenario

We use the following reference scenario to identify some key architectural requirements that have to be satisfied in order to support context awareness in ANs, validate our design, and evaluate our prototype implementation.

Two friends, each equipped with PDAs and various other network devices in their respective PANs, meet at a station. Their PANs identify each other, and based on their respective policies, decide to compose, jointly forming an ambient network. They start to share each other's playlists, music files and podcasts on their MP3 players. A hotspot at the railway station allows the two friends' ambient networks to compose with the rail company's public network, providing access to other online services such as ticketing, arrival and departure updates, and other travel-related information and services.

As their train arrives at the station, the announcement on the arrivals board is accompanied by a gentle audio message sent directly to them through their MP3 players. (It is interesting to note that their MP3 players are only connected to their PANs via Bluetooth and not directly to the 802.11 wireless LAN within the station.) As the two friends board, their e-tickets are validated, and their ANs compose with the train's ambient network.

While the friends wait for their train to depart, a third-party "service provider" car is connected to the train. Aside from the usual physical items that passengers may purchase, this car also offers both audio and video on-demand, and is equipped with satellite links offering continuous Internet access as well as satellite TV channels. The new car's ambient network composes with the rest of the train's ambient network so that passengers may access the new offered services. A member of the railway staff then boards the train to do a final inspection, reporting his findings to a colleague in the control room through a video link on his portable device. As soon he disembarks, he gives his "thumbs-up" to the control room to allow the train to proceed on its journey.

The train leaves the station and the friends' journey begins. They continue to share each other's material and browse through the offerings of the service provider car. One friend decides to access her office email, while the other views a live news feed on her PDA via streaming video. A dynamically updated journey map is continuously displayed on their PDAs by the train's AN throughout the journey.

As they approach their destination, they are again gently alerted by a message sent through the train's public address system as well as to their MP3 players. They both disembark at the railway station to wait for their connecting ride. Within the station, there are hotspots provided by multiple Internet service providers as well as by the rail company. The person who used to be accessing email on the train now composes with the network that provides best-effort quality of service, while her friend, who was watching a video feed, composes with the network that provides a premium service capable of supporting the video stream. They also compose with the rail company's AN, which provides them unobtrusive updates on the status of their onward journey.

3.2 Design Requirements for ContextWare

The scenario depicted provides some examples of context use:

– The location of the users and their PANs; the location of the train network as it approaches the station and continues with its journey; the location,

coverage and access availability of the hotspots at the train stations; motion information such as their speed and direction, or predicted movement
- The identities of the users and network IDs of their PANs (ambient networks), as well as the network or operator IDs of the train ambient networks and ISPs at the train stations; the composition status of these ambient networks
- The applications run by the users and the high-level activities and intentions behind these applications
- The capabilities and services offered by the devices in the PANs; their ability to handle certain multimedia streams in certain formats; the capability of nodes or "middleboxes" within the network to process these multimedia streams; other services available within the ambient networks
- The QoS offered by all networks involved, including the PANs, the train ambient network, and third-party Internet providers at the destination train station
- The security and privacy level required by the users and their application streams; the levels of security offered by the ambient networks

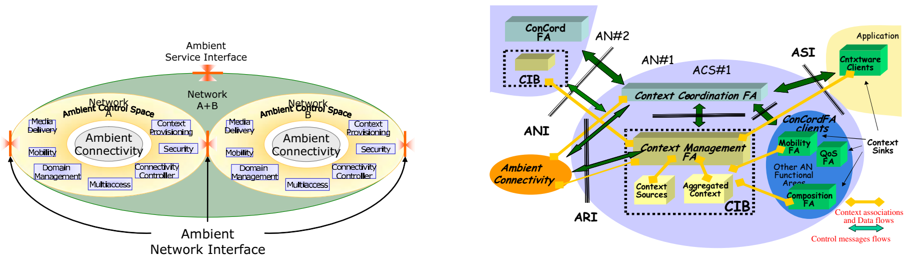
One obvious and fundamental requirement suggested by this scenario is the need for an integrated architectural solution to the management of context within the network, i.e. ContextWare. Beyond this high-level requirement, following more specific functional requirements for ContextWare have been identified [4]:

1. *Context-sensitive communications.* Within ambient networks the exchange of information between entities takes place in a manner appropriate to the situation of the entities and components of the communication environment, and this process should be coordinated based on the requirements of these entities. Context-sensitive communications (CSC) consist of the protocols governing the different phases of information exchange, the conceptual channel needed to convey the information itself, and a conceptual channel containing meta-information and control governing the exchange.
2. *Cross-layer context handling.* ContextWare should be able to handle context originating from different system layers, from applications all the way down to session, network or link layers. Similarly, relevant context information may also be consumed either by network entities or applications.
3. *Context information base.* ContextWare should make context information accessible to clients in a simple yet efficient way that abstracts or hides the internal implementation details from the context consumer. Although it may be seen by external context clients as a logical context store, internally the CIB manages, processes, aggregates, stores and delivers context in a highly dynamic, efficient and distributed way.
4. *Context associations.* ContextWare should be able to form and manage temporary associations between context sources and sinks, characterized by channel establishment and the exchange of context information between the entities.

5. *Context level agreements.* Context level agreements refer to the negotiation, establishment and enforcement of agreements and policies governing context information that is shared between parties.
6. *Support for external applications.* In order to allow external applications to access ContextWare functionality within the ACS, the interface of the ACS to the “outside world” – the Ambient Service Interface or ASI – should support queries, requests and replies from context client applications directed towards ContextWare functionality within the ACS.

3.3 ContextWare Architecture

There are two main Functional Areas (FAs) within the ContextWare architecture, namely the Context Coordination (ConCord) FA and the Context Management (CM) FA, shown in Fig 1b. The Context Management FA is further enclosed within a logical entity called the Context Information Base (CIB).



(a) (b)

Fig. 1. (a) The Ambient Control Space. (b) ContextWare architecture.

The ConCord FA is the first point of contact for any context client, which may be another functional area within the Ambient Control Space (see Figure 1b), or an external application that requires context information. Clients request context information and negotiate the type and quality of information that they will receive. ConCord manages the establishment and enforcement of policies and resolves potential conflicts governing the kind and quality of context that are exchanged between parties, such as between ACS FAs, the ACS and an external client, or two ANs, using Context Level Agreements (CLAs).

The CM FA manages operations such as the collection, modeling and dissemination of context information within an AN. It is also responsible for scheduling and monitoring interactions between context sources and sinks, as well as allocating resources (such as communication channels) for such interaction. The CM FA also manages the processing, manipulation and aggregation of context information from raw form directly from context sources into the information required by the context client, as agreed upon in the CLA.

The CM FA, context sources, sinks, and processors/aggregators collectively form the Context Information Base, or CIB. The CIB is a logical construct that appears as a distributed repository for context information within the AN. That is, from an external entity, the CIB appears as a single logical context store, abstracting or hiding the detailed workings of the functional components within.

4 Prototype

A prototype was developed jointly by the Budapest University of Technology and Economics (BUTE) and the University College London (UCL), with technical input and guidance from the other partners of the Ambient Networks project. This prototype demonstrates the management of context and other ContextWare functionality, particularly in supporting the dynamic adaptation of a network service called an Active Virtual Pipe, or AVP, using context information from various sources within the network including network composition events within the ACS.

4.1 Programmable Approach

Our current ContextWare prototype implements a subset of the architectural components and functions specified in Sect. 3.3. ContextWare components are dynamically deployed and executed within the ambient network prototype using an active and programmable networking [6, 7] approach, implemented on the DINA active networking platform [8].

ContextWare Active Components. We implemented two general classes of ContextWare active components, namely Context Monitoring Systems (CMS), which are basically software-based context sensors and aggregators, and the Context Manager (CM).

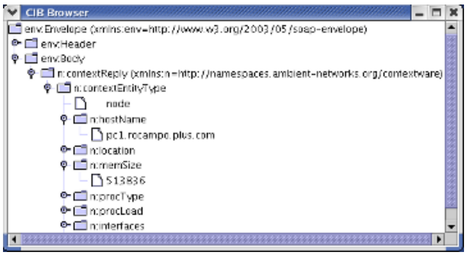
The main functionality implemented by the CM prototype at the moment consists of CMS registration and limited query support. When CMS objects are deployed and instantiated, they register information about themselves, the information they can provide, and the IP address and UDP ports where they may be queried, with the CM. Typically, a context client such as an application or a functional area (FA) within the ACS would query the CM for CMS objects that can provide a certain type of information.

CMS objects obtain context information by interfacing with the Composition Monitoring functions [9] within the ACS and subscribing to context events. They also obtain network-related context from other sources, including the SNMP Management Information Base (MIB) or other low-level sources of network information. In our implementation, we initially limited ourselves to three main classes of CMS: one that receives context event notifications from the composition management application, another that monitors network QoS, and a third that obtains node- and device-related context information. (We will henceforth use the term “sensor” to refer to CMS objects.) The details of these sensor classes are as follows:

P2P context event sensor. The P2P context sensor listens for events from a peer-to-peer overlay management platform (details of which may be found in [9]) that manages network composition. Our current implementation of this class uses SNMP traps defined through a MIB object to support event notifications to context clients. Some examples of context events monitored by this class of sensor are: `newOverlayMemberJoined`, `overlayMemberLost`, `becomeSuperPeer`, and `superPeerLost`.

QoS context sensor. This class of sensor provides information on the QoS state of an ambient network. Our prototype uses a subclass that dynamically monitors bandwidth utilization on an interface, or more specifically, the traffic traversing a particular Linux IP chain [10]. Unlike the other two sensor classes whose implementations were written fully in Java, the bulk of the code for this sensor was written in C, for two major reasons. The first reason was that sensors of this class required low-level access to node services and resources not normally accessible via Java, such as high-resolution timers and per-chain packet and byte counters. A second reason was that performance and code execution latency were seen as a crucial factors in generating real-time QoS measurements.

Node and device context sensor. This class of sensor provides information on node-related context, such as its host name, processor type and load, memory size, available interfaces, display capabilities, and others. In most cases it obtains information from the SNMP MIB through the DINA platform. The sensor can also provide information on the geographic location of the host, although the data currently supplied by the sensor is only a simulated location. Figure 2 shows part of a reply to a context query sent to a node sensor; the information shown in the diagram is displayed through the *CIB Browser*, a small application we have developed for debugging and diagnostic purposes.



```
env:Envelope (xmlns:env=http://www.w3.org/2003/05/soap-envelope)
  env:Header
  env:Body
    n:contextReply (xmlns:n=http://namespaces.ambient-networks.org/contextware)
      n:contextEntityType
        node
        n:hostName
          pc1.rocampo.plus.com
        n:location
        n:memSize
          513836
        n:procType
        n:procLoad
        n:interfaces
```

Fig. 2. Fragment of a context reply from a sensor object, as viewed in the CIB browser

Context replies from node sensors are typically formatted in Extensible Markup Language (XML) [11] although other proprietary formats are also supported. It should be noted that although the sensor reply shown in Fig. 2 implies the use of Simple Object Access Protocol (SOAP) [12] message formatting, the protocol itself is not supported in our current implementation. In general, all of these sensors support event notifications to context clients, however, due to the nature of the context information they provide, node sensors are typically used in query-response mode. On the other hand, due to the highly dynamic nature of network QoS, clients of QoS sensors either subscribe to continuous information streams, or request notification when there are significant context changes such as congestion events or when traffic levels exceed a certain threshold.

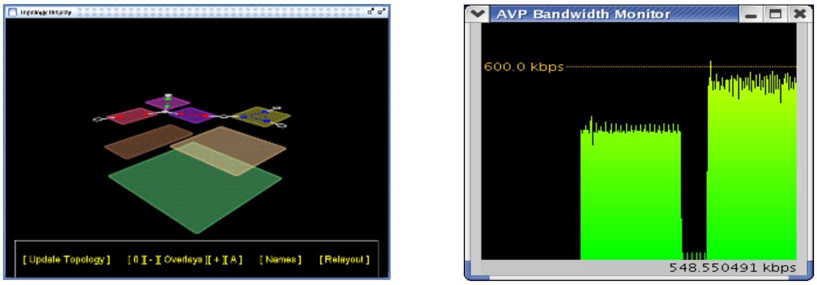
Resource Minibrokers. In some cases, such as with QoS context sensors, it was not practical to implement the sensor fully as a DINA active packet. In cases like these we designed and implemented *minibrokers* to provide access to low-level node resources. Unlike DINA's brokers [8], which provide a relatively wide range of services and which are an integral part of the DINA active platform, our minibrokers are relatively lightweight and implemented for very specific purposes, provide access to only a narrow range of low-level resources or services, and are generally not tighly integrated with the DINA platform. For example, for QoS sensors, a minibroker provided access to byte and packet counters through mechanisms provided by Linux `iptables` as well as time measurements with microsecond resolution using C's `gettimeofday()`. The other half of the sensor was implemented as the DINA active packet; when this other half was deployed to the node, it would be able to interface with the appropriate minibroker. As will be seen in the next section, the minibroker approach is also used in some cases where services such as AVPs are dynamically instantiated and adapted using DINA active packets.

5 Validation of the Prototype

As a means of partially validating our prototype, we set up an experimental testbed consisting of six notebook computers, each equipped with wired Ethernet and 802.11 wireless network interfaces. All ran the integrated prototype on the Linux operating system. We revisited the scenario in Sect. 3.1 and mapped the devices in our testbed to some of the scenario actors: two of the computers represented the respective PAN ambient networks of the two friends, one computer represented the railway staff's AN, another two represented the ambient networks of the train passenger car and the service provider car, and finally one computer represented the rail company's hotspot at the railway station.

We then walked through selected parts of the scenario using our testbed. The ANs of the two passengers first composed with each other based on their AN identifiers and their respective composition policies. We then simulated their joining the train's ambient network by switching the two passengers' 802.11 ESSIDs to that of the train's 802.11 ESSID; as soon as they detected each other, the composed AN of the two passengers composed with the train's AN. The interconnection of the passenger car, the service provider car, and the train station network were simulated in sequence using Ethernet (wired) connections. Finally, we simulated the railway staff boarding and disembarking from the train using the same ESSID switching technique.

Figure 3a shows the state of the testbed from the point of view of the peer-to-peer management application [9], at that point in the scenario where the two passengers and the train staff are on the train while the train is still in the rail station. The application shows each ambient network as a node in the graph and graphically illustrates the topology, hierarchical structure and composition state of the AN management overlay.



(a) (b)

Fig. 3. (a) Topological view of the peer-to-peer management overlay. (b) AVP bandwidth adaptation in response to context change.

5.1 AVP Deployment and Self-adaptation

In ANs, management-layer composition is also an integral component of network composition. As a vehicle for management-layer composition, *Ambient Virtual Pipes* (AVPs) provide secure, QoS-assured and adaptive management service overlays that are autonomically built on top of the underlying composed P2P AN hierarchy [9]. *AVP Managers* subscribe to ContextWare components and use context information for dynamic provisioning and self-adaptation of the management overlay.

During each composition, the election of a management superpeer [9], the identities of the composing ANs, and overlay join/leave events were sensed as context, triggering the deployment or self-adaptation of AVPs interconnecting the management peers. When the railway staff boards the train and composes with the train AN in the reference scenario, the associated context events trigger policies on a superpeer that determine the amount of bandwidth to be allocated to the AVP. As the railway staff member disembarks from the train and composes directly with the railway company's AN, this change in context, i.e., movement from an AN owned by a third-party operator to his home network, triggers an adaptive reallocation (increase) in the AVP's bandwidth. The graph in Fig. 3b shows the comparative bandwidths allocated to the AVP connected to the railway staff member's AN as he moves from one AN domain to another. The graph is provided in real-time by a management application called the *AVP Bandwidth Monitor* and the data represents the amount of actual video traffic transmitted through the AVP. The data is provided to this management application by QoS sensors we discussed in Sect. 4.1; this is another concrete example of an application acting as a client of ContextWare.

6 Conclusion and Future Work

We have discussed ContextWare's role in providing an integrated infrastructure for sensing, processing and distributing context within ambient networks, thereby making ANs *context-aware networks*. This context-aware functionality plays an important role in enabling ANs to deal with device and network heterogeneity, adapting to highly mobile and variable network conditions, and providing end-to-end services that adapt to user needs and ambient conditions. We have outlined ContextWare's design and architecture, and likewise presented the architectural design of a prototype. Our current implementation uses an active networking approach as we believe this allows ContextWare components to be flexibly and dynamically provisioned, on-demand, within ambient networks.

Using our reference scenario, we have demonstrated how ContextWare plays a mediating role within the ACS between a context source (the composition and overlay management application) and a context client (the AVP service manager). We plan to further develop the prototype so that it can also demonstrate a reversal of roles, that is, we intend to show context information obtained from ambient network services and applications, this time acting as context sources, driving or influencing the composition process in a context-aware way. For instance, the QoS needs of an application being executed by a user would dictate a decision to compose with one or more ANs, taking into account the QoS, cost, and other contextual information shared by these ambient networks.

Acknowledgment. This paper describes work undertaken in the context of the Ambient Networks - Information Society Technologies (IST) 507134 project, which is partially funded by the Commission of the European Union. We would like to acknowledge the work done by R. Szabo, S. Csaba, M. Erdei, P. Kersch, A. Wágner and other colleagues from BUTE for their work on the P2P platform, as well as the contributions and feedback of the other partners from the AN project. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the Ambient Networks project. Roel Ocampo acknowledges support from the Doctoral Studies Fellowship program of the University of the Philippines.

References

1. K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten and J-C. Burgelman. Scenarios for Ambient Intelligence in 2010. Final report of the IST Advisory Group, February 2001. Available at ftp://ftp.cordis.lu/pub/ist/docs/istagscenarios2010.pdf
2. N. Niebert, H. Flinck, R. Hancock, H. Karl and C. Prehofer. Ambient Networks - Research for Communication Networks Beyond 3G. *13th IST Mobile and Wireless Communications Summit 2004*, Lyon, June 2004.

3. B. Busropan, F. Pittman, S. Ruffino, O. Karasti, J. Gebert, T. Koshimizu, D. Moro, B. Ohlman, N. Papadoglou, A. Scheider, W. Speltacker, V. Typpö and S. Svaet. Ambient Network Scenario, Requirements and Draft Concepts. *Deliverable Report D-1-2, Ambient Networks Project*, October 2004. Document number IST-2002-507134-AN/WP1/D02, available at http://www.ambient-networks.org/publications/AN-D1-2%20public.pdf
4. A. Jonsson, R. Giaffreda, M. Barachi, R. Glitho, F. Belqasmi, M. Smirnov, M. Kleis, C. Reichert, A. Karmouch, M. Khedr, A. Karlsson, H. Laamanen, H. Helin, A. Galis, R. Ocampo and J. Zhang. Ambient Networks ContextWare: First Paper on Context-Aware Networks. *Deliverable Report D-6-1, Ambient Networks Project*, January 2005. Document number IST-2002-507134-AN/WP6/D61, available at http://www.ambient-networks.org/publications/D6-1_PU.pdf
5. A. Karmouch, A. Galis, R. Giaffreda, T. Kanter, A. Jonsson, A. Karlsson, R. Glitho, M. Smirnov, M. Kleis, C. Reichert, A. Tan, M. Khedr, N. Samaan, H. Laamanen, M. El Barachi and J. Dang. Contextware Research Challenges in Ambient Networks. *1st International Workshop on Mobility Aware Technologies and Applications*, October 2004.
6. D. L. Tennenhouse and D. J. Wetherall. Towards an Active Network Architecture. *Computer Communication Review*, 26(2), April 1996.
7. A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente and D. Villela. A Survey of Programmable Networks. *ACM SIGCOMM Computer Communication Review*, 24(2), April 1999.
8. D. Raz and Y. Shavitt. An Active Network Approach for Efficient Network Management. *International Working Conference on Active Networks 1999 (IWAN99)*, Berlin, Germany, July 1999.
9. A. Galis, L. Cheng, M. Brunner, S. Schuetz, G. Nunzi, E. Asmare, J. Nielsen, A. Gunnar, H. Abrahamsson, R. Szabo, S. Csaba, M. Erdei, P. Kersch, Z. Kis, B. Kovács, R. Stadler, A. Wágner, K. Molnár, A. Gonzalez, G. Molnar, J. Andres, and M. Callejo. Ambient Network Management - Technologies and Strategies. *Deliverable Report D-8-1, Ambient Networks Project*, December 2004. Document number IST-2002-507134-AN/D8-1, available at http://www.ambient-networks.org/publications/D8-1_PU.pdf
10. B. Hubert, T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. Schroeder, J. Spaans, P. Larroy (editors). Linux Advanced Routing and Traffic Control. http://www.lartc.org.
11. T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau (editors). Extensible Markup Language 1.1. *W3C Recommendation 04 February 2004*, http://www.w3.org/TR/xml11
12. M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Nielsen (editors). Simple Object Access Protocol (SOAP) 1.2. *W3C Recommendation 24 June 2003*, http://www.w3.org/TR/soap12

Fixed Mobile Convergence: 3 Words, Many Perspectives

Italo Milanese and Carlo Alberto Licciardi

Telecom Italia Lab, via Reiss Romoli 274, 10148 Torino, Italy
italo.milanese@telecomitalia.it,
carloalberto.licciardi@telecomitalia.it

Abstract. This article describes the TILAB view about the fixed mobile convergence. Starting from the different perspectives of all the possible actors in the Telecommunication arena, we introduce all the possible level of convergence and present the TILAB experience about the Service Creation Layer. This is considered a key point to optimize the exploitation of all the opportunities coming from the new categories of services which can be developed at the convergence of the mobile and the fixed networks.

Keywords: Fixed Mobile Convergence (FMC), Service Creation Environment (SCE), Service Delivery Platform (SDP), Session Initiation Protocol (SIP).

1 Introduction

After some years of independence between the fixed and mobile businesses, with the mobile and the fixed business separated in order to maximize the mobile growth rate and the possible revenues from the mobile market, three facts recently changed the situation:

- The number of mobile lines overcame the number of fixed lines (in Italy the revenues overcome of mobile against the fixed happened in 2003);
- The fixed to mobile traffic erosion, where the usage of the mobile terminal also in indoor situation is becoming more and more popular...
- The process of fixed to mobile line substitution, with the progressive loss of the subscription fee for the incumbent fixed operators.

Together with these market signals, some technical trends progressively appeared on the scenario of the Telecommunication (TLC) of the new century, giving a new flavor to the relationship between mobile and fixed networks and relating operators: Here are the most interesting:

- The progressive “IP-zation” of the TLC traffic, starting from data and including also voice, with the growing diffusion of Voice over IP (VoIP), often under the form of models à la Skype;
- The target potential architectural coincidence between the mobile and fixed networks towards IP Multimedia Subsystem (IMS), with Session Initiation Protocol (SIP) in the role of session control reference protocol;

- The personalization of the communication , where the calls and the sessions are going to be set up no more between terminals and circuits but between persons who can use different terminals and different media, independently from the used networks. A linked point is the willingness of the user to be reachable with no regard to the called phone number.
- the growth of the available bandwidth for the wireless geographical access is minor than what expected in the last years, still providing to the fixed networks a significant advantage in terms of access to Broad Band (BB) services. Wireless Local Area Networks (WLAN) and BlueTooth popularity (WiMax in the future) seems moreover particularly interesting for expanding the BB access to IP based services and potentially also for voice based services.

All those signals marked the importance for the TLC operators to set up synergies between the fixed and the mobile networks and services, therefore many of the big TLC players tried recently to set up either internal re-arrangements (e.g. Telecom Italia and TIM, France Telecom and Orange etc) or external partnerships (e.g. BT and Vodafone for BluePhone, with the set up of a Mobile Virtual Network Operator (MVNO) license).

2 The FMC Today: Some Different Points of View

In the current scenario about the F/M convergence the different actors can have very different points of view.

The **fixed-only operator**'s first objective is of course defending himself from the fixed-mobile substitution and from the traffic erosion, with both threats coming from a growing usage of the mobile phones also in the indoor locations or in general where an additional fixed line could be available. The **mobile-only operator** is currently in a good position, with growing figures for the traffic and for the revenues, but the general opinion is that the market saturation for the mobile is going to approach and in some case it's already here. We think that the **fixed AND mobile operator** is in the best position, with the possibility to exploit the advantages of the mobile and of the fixed part: our opinion is demonstrated by the fact that so many operators are currently trying to extend their business to the other counterpart.

3 Many Levels of Integration Between Fixed and Mobile Networks

There are many ways to intend the FMC, from the hybrid terminals adoption till to the billing procedure integration, at the opposite extremes. In this chapter we briefly introduce some of the possible kinds of integration between fixed and mobile.

3.1 Terminal Convergence

The bottom level of convergence implies the adoption of multi radio terminals, which typically allow the usage of GSM/GPRS/UMTS (mobile) interfaces in conjunction with other networks like WLAN or Bluetooth.

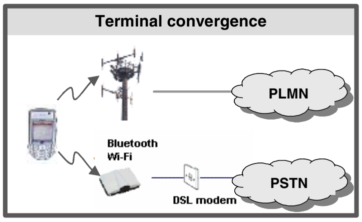


Fig. 1. Terminal Convergence

In this case the WiFi or Bluetooth wireless interfaces are linked to fixed networks in a local environment: the dual mode terminal can camp either in the mobile or in the fixed network depending on the chosen configuration. No kind of integration is here set up among the networks, which remain completely separated; just a service/commercial agreement is needed in order to charge and bill the service usage in a proper way.

3.2 Network Control Convergence

From the point of view of the technology evolution this is probably the deepest case of convergence. Here the integration is set up at a network control level, where fixed and mobile networks share not only the same "signaling" protocol (SIP for the IP sessions control) but also the same IMS architecture principles, with similar network elements in both cases. The relevance of IMS and SIP as technological enablers is proved also by the dimension of effort in the standardization fora, in the mobile (3GPP UMTS rel. 5 and following) and in the fixed area (e.g. TISPAN rel. 1.0 based on IMS rel.6.0).

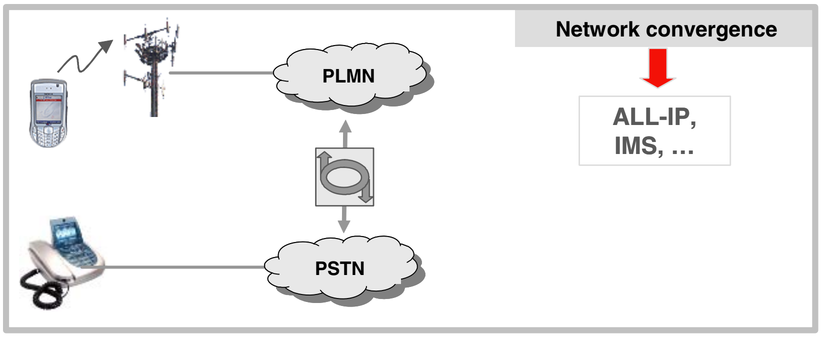


Fig. 2. Network Convergence

This convergence fits especially for session based services but taking into account the progressive "packetisation" of the whole TLC traffic, this approach can be extended to the basic voice based services. Presence based features, registration of the customers to the network with the possibility of enjoying the services independently from the used terminal, possible mixing of multimedia into the convergent communication are all examples of evolutions enabled by the SIP based network control. In addition to the SIP based direction, many other options are currently under analysis or under development, with different levels of standardization and maturity.

3.3 Management/Logistic

The convergence at this level often appears as less attractive, in particular for the apparent minor importance of the concerned technology innovation. From the customer point of view the value proposition of the possible advantages is at the opposite very high. Some typical examples are the possibility to receive a single bill for the fixed and for the mobile subscription or to access to a single contact point for all the needed interaction with the mobile or fixed service (new subscriptions, complaints, faults communication, request of information about bills, traffic and so on). From the point of view of the operators a lot of savings can come moreover from the optimization of the activities relating to operations and management of both the networks and in the operative organization for all the activities on the field (repairs, sales, advertising, marketing etc).

3.4 Service Execution and Delivery

The possibility to set up a fast creation, prototyping and execution of new services relies on an execution layer working effectively with the Service Creation Environment (SCE), in order to make the phase of service delivery as fast as possible. This layer can be realized by one or more execution environments (application servers or Service Logic Execution Environment (SLEE)) based on different technologies. Here the interaction between services or between services components acting on different technologies is also performed, including the resources adaptation functions.

In the most innovative versions of the SDP (Service Delivery Platform) architectures, the Service Exposure Layer allows the openness with secure and controlled interfaces to third parties for setting up business relationships (about co-use of service components or services) with the TLC network operators. In this context the most used technology is based on Web Services, with Parlay X interfaces as defined by OSA Parlay Group.

3.5 Service Creation

On the top of service execution environment and relating capabilities the key point is the the complete control of a dedicated tool for the service creation. In a specific section of the paper this subject will be explained with more detail, introducing also the TILAB approach to the SCE for convergent services. The most relevant feature of SCE for the service provider is to speed up the creating, prototyping of new services by linking existing network resources and enabling easy testing of new metaphors of services, allowing service providers to develop new services in an effective way without the mediation of industrial service developers.

4 An Honest Look at the Reality

About the FMC possible success or failure, it is possible to register several different opinions in the specialized magazines, among the analysts and in general in the TLC world. Anyway, many facts show that something new is going to happen, even if probably there will not be a revolution in the short-medium term.

Looking at the commercial availability of the multi-radio terminals, in march 2005 there were about 8 announced/available terminals (smart phones and Personal Digital Assistant (PDA)) with multi radio interface (GSM/GPRS or CDMA and WiFi and BlueTooth). Among them 4 were announced, as commercially available, with different levels of reliability. In most cases the VoIP support on SIP clients and the battery duration were the most severe problems. None of the available models appeared to be as the potential absolute winner.

In the meanwhile in the application field no more than six or seven real convergent services have been proposed or announced worldwide by the biggest operators, for the residential and for the business customers. For all of them the core value proposition is based on a flexible usage of the fixed and the mobile numbers or terminals between the indoor and the outdoors environments, with a particular attention to the voice services. About these voice services the approach of the mobile operators ("discount based " push to a universal usage of the mobile terminal, even in indoor environment) of course differs from the strategy of the fixed operators (VoIP and VoWLAN dual mode terminals for optimizing the mobile usage).

According to this scenario the most known initiatives set up by most important operators are Unlicensed Mobile Access (UMA) and FMC Association (FMCA), which from different point view have the same objective of attacking the voice market in the contested field of the indoor areas.

FMCA,(set up by BT in 2004 with about ten other fixed and mobile operators) has the purpose to push forward fixed-mobile convergence into Telco market, with sharing technical solution pruning criteria and trial results, advising and influencing device and infrastructure providers, define technical requirements for devices and Access Points, accelerate standardization.

UMA (founded by solution providers and mobile operators) proposes technical open specifications for accessing to GSM and GPRS services through dual mode devices over unlicensed spectrum technologies (Wifi, Bluetooth and more…), addressing standards about Interoperability & handover between Wifi and GSM/GPRS both for circuit and packet services, Security (authentication, integrity and confidentiality) and pushing for the concept of "One phone number" (with the network control owned by Mobile Network Operator (MNO)). It requires an interworking element called UMA Network Controller (UNC) in the access network and the adoption of UMA complaint dual-mode terminals.

5 New Opportunities in a Complex Scenario

So far what it seems to be completely missing is a commercial proposal of services which can really exploit a new generation of fixed-mobile services from a different perspective than simple voice services. From this point of view a very broad range of possible new categories of services appears still to be exploited and the most promising aspect is that can be considered as additional to the existing voice market. The idea, in particular for the fixed AND mobile operators, is to leverage the strengths of both networks and introduces a new set of Value Added Services, using all the available new technology opportunities.

In this context, the fixed component can exploit the cheapness of the broadband in the access, the perceived bigger safety of using fixed terminals at home, the bigger usability of the PC for the fruition of a large amount of contents at home or in indoor situations. The mobile component can on the other hand exploit the fact that you *de facto* carry always the mobile terminal with you and therefore it allows a permanent contact with the network and the continuous possibility to activate or control any kind of services, whose enjoyment can also be postponed to a further moment where the customer can use a fixed line.

Presence, context awareness, centralized and manageable customer profiles, localization, wireless BB access (WiFi and Blue Tooth in particular), common (fixed/mobile environments) contents, SIP based control architectures, and in particular a powerful and flexible SCE are all components of a technological puzzle which can open new opportunities for the TLC operators.

The real challenge is to be sure, here and now, about the correct mixture of technology and innovation that can provide a set of service with the right features to be successful in the FMC area. A strategy based on many tries appears as the most reasonable, therefore a powerful and flexible tool for fast creation, prototyping and executing new metaphors of service in our opinion is a key point for an operator which wants to exploit all the opportunities in the FMC area.

The next chapters describe the TILAB view about the design, the development and the usage of a new generation SCE, based on StarSIP, the TILAB SIP based platform which is currently used by Telecom Italia for all the SIP based services.

6 An Advanced Service Creation and Execution Environment

The differentiation and the competition between TLC operators is mainly based on innovation, attractive and easy use of new services, as well as the ability to quickly adapt services to the requirements of the market and of the customers. It's therefore really important to make use of tools that simplify and fasten the creation and deployment of services. The starting point to define a services creation system is to analyze the service life cycle, composed by the following phases:

- **Ideation**: handled by the service provider marketing, it follows a phase of user needs analysis. It ends with the definition of service requirements or basic service concepts.
- **Development**: requirements are translated in service logic using programming language and IDE (Integrated Development Environment), SDK (Software Development Kit) or graphical programming environments.
- **Simulation and testing**: tests are performed in a simulate environment to verify the right implementation and to identify interactions with existing services.

- **Deployment**: deployment into the network can include a software configuration phase of network elements and user clients (software upgrade on a terminal).
- **Provisioning**: the service is made available for subscribed users (also implicit ones). This phase can be ended with the service advertising to users.

The SCE must provide high level features that allow reusing efficiently the huge catalogue of existing high-level or more basic building-blocks existing either at the platform side or directly through the web. Since all these bricks provide a number of nice and rich capabilities, it is foreseen that providing new services will consist of combining, orchestrating or controlling elementary service components, network features and third party services, taking also into account other sources of information such as user's profile and preferences, context etc.

6.1 Service Execution Models for the Service Platform

Service execution should ease integration of heterogeneous fixed and mobile resources (SIP Application Server, Messaging server, Conferencing server) by means of abstraction of resources and protocols. The execution model should allow an execution engine to host libraries of building components which hide the complexity of underlying resources.

Also, the advanced service execution model will require the service to be distributed among the service platform and the mobile terminal since the service will leverage on components available on both sides. Once designed, the service must be delivered in an easy and seamless way to the execution engines to become executable.

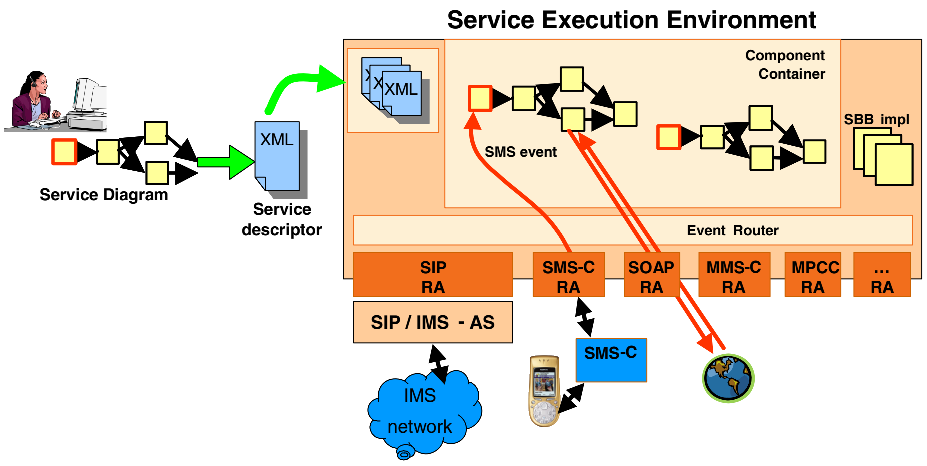


Fig. 3. Service creation, execution and interaction with other technologies

Moreover, service delivery defines mechanism to verify that all components required to execute a service are available on the execution engine, and methods to find and install missing components. The maintenance of different versions of a service and changes in its configuration should be supported by service delivery. TILAB developed an event-based Service Logic Execution Environment (StarSLEE) and the corresponding SCE (StarSCE).

The StarSLEE engine supports the execution of event-based service logics that span different application platforms and network (fixed and mobile). The engine is modeled following the emerging JAIN (Java Advanced Intelligent Network) Service Logic Execution Environment (JAIN SLEE) framework and can be easily connected to different application platforms or network elements via a Resource Adaptor (RA) mechanism. RAs to different external platforms and elements have been already developed.

The JAIN SLEE specification defines how TLC services can be built, managed and executed. It has been designed explicitly to enable vendor products to address carrier grade requirements in a standard call-signaling environment. JAIN SLEE provides a standard programming model that can be used by the large Java developer community. The programming model has been designed to simplify the work of the application developer, to eliminate common programmer errors and to ensure that robust services can be rapidly developed. JAIN is a Java technology, therefore all the standard Java APIs (Application Programming Interfaces) can be leveraged in signaling applications. The component model specifies how service logic is to be built, packaged, receives outside events and executes. This SLEE framework enables multiple components from different authors and networks to collaborate. RAs are responsible for the communication with a particular resource that is external to the JAIN SLEE programming model, communicating with the event routing logic in the SLEE implementation and providing the programmer with a suitable API.

6.2 StarSCE

StarSCE, based on the industry standard Eclipse IDE, allows the development of advanced services by simply using graphic composition of a portfolio of atomic services, called “Service Building Blocks” (SBB). Using this tool, TILAB developed many SBBs to enable the fast development of integrated fixed and mobile services. The most important are: Call Control (using SIP and OSA interfaces), Multiparty Call Control, Presence component, Instant Messaging and Presence (using SIP), SMS and MMS messaging, Geo-localization, Generic Web Services invocation (weather forecast, stock quote, yellow pages etc) and non-functional blocks to control the service work-flow (e.g. for, loop, switch, timer, join...).

The execution engine is built as a Java framework which implements an event-based (publish-subscribe) model. The services developed using this tool are described by an XML (eXtensible Markup Language) defined by TILAB (also known as service composition language). The language is generated by the Service Creation Tool in a graphical way, and is executed on the execution engine, allowing run time composition of elementary components. The designer tool is fully graphic and does not require any Java programming skills for the development of composed services starting from the portfolio of available service building blocks.

6.3 Service Creation Environment Tool (IDE and Graphical Designer)

SCE-IDE is a graphical environment which provides tools for programming and services communication design. SCE-IDE contains two tools:

- Service Development Suite
- Service Composer

Service Development Suite (Java programmers oriented) offers an advanced IDE with:

- A graphical editor with Java advanced functionalities (code completion, syntax highlight, etc.)
- A Java code graphical debugger
- An instance of StarSIP/StarSLEE platform to test the developed services, (to debug the code of the service)

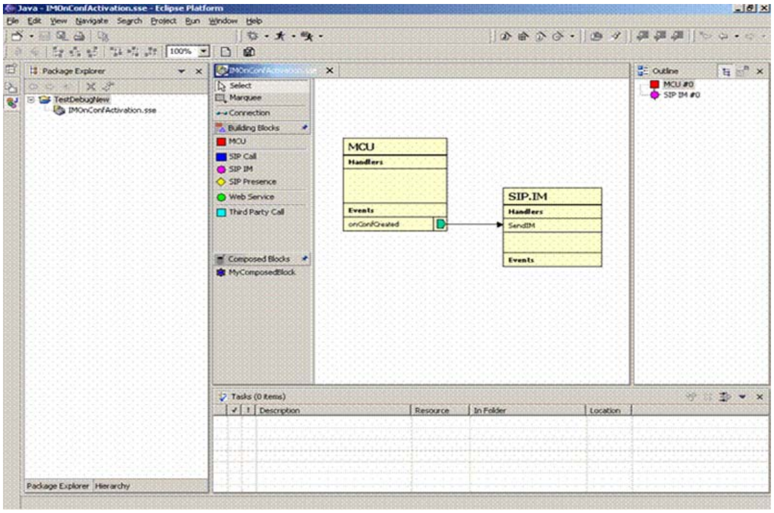


Fig. 4. SCE-Service composer tool

“Wizards” are also available to maximize the automatization of the programming, creating for example in automatic and customized way a new “StarSIP & StarSLEE Service Project”, with configuration files and all classes needed to develop the new service starting from a template. The execution environment can be enriched of new features (e.g. SBB and RA), becoming very flexible and easily adaptable to new technologies for supporting the needs of service providers and customers. The programmer can find in one single environment all the necessary tools for services development (libraries, documentation, editor, debugger, test platform).

The **Service Composer** is an environment for graphical design of services, by basic functional blocks composition. Basic functional blocks composition occurs through a graphical interface for editing the diagram showing the service logic. Functional blocks used for service design are selected from an icons library, by ‘drag and drop’ mechanism, and customization parameters of each basic block can be graphically edited.

Figure 4 represents a screenshot of the environment developed by TILAB. The palette on the left-hand-side contains the elementary components (SBB) that can be composed to provide added value services. Components are dragged on the working area at the right and are connected to build the new value added services.

6.4 Building a Service with the StarSCE Graphical Designer (Example)

This section describes an example of videoconference service developed by using the service creation designer. The service allows a customer to create a multimedia/multiparty conference session among a group of people (“Myfriends”) previously configured by the user, by sending an SMS “call Myfriends”.

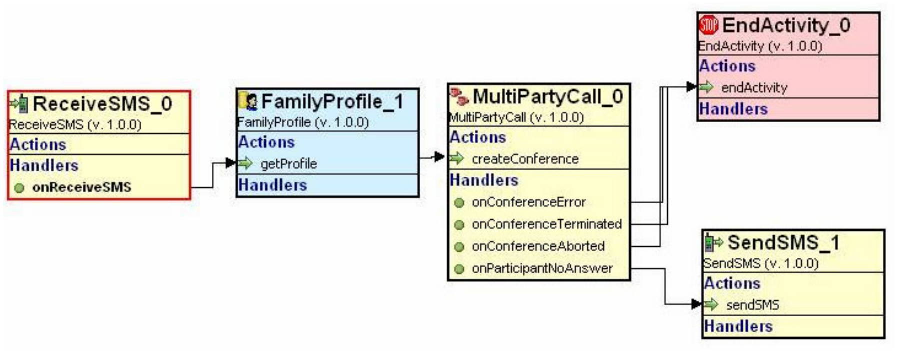


Fig. 5. Service diagrams for the multi-conference service

The group can consist of fixed and mobile users, which can be equipped with heterogeneous terminals (fixed, mobile, broadband video or only voice enabled). If one or more users invited to attend the conference do not answer he/they are notified of the invitation by an SMS and they can join the conference at a later stage. Figure 5 shows the service diagram built with the Service Composer editor. The functional blocks which compose the service are:

- **MultipartyCall** (based on a Multipoint Conference Unit): a functional element to start/end and handle a Audio-Video Conference among several participants.
- **ReceiveSMS & SendSMS**: the block enables to receive/send an SMS from/to the service logic. The SMS is sent from the user to activate the conference with his friends, and it is sent by the service logic to the user(s) which does (do) not answer to the conference invitation. It’s useful to join the conference at a more convenient time.
- **FamilyProfile:** this block will query the user profile to translate Myfriends (the group name) in the list of users to be invited at the conference.

The service designer assembles the components linking them according to the service requirements and configures the parameters for individual blocks (mainly defining the handlers and the actions); in other words it personalized the SBB to fulfill service requirements. The designer tool will produce an XML file to be deployed on the service execution environment to make the service executable (e.g. it can be invoked by the user).

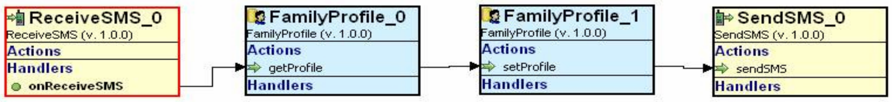


Fig. 6. Service diagram for the group management by SMS

Going back to the service concept, the user can customize the group (adding a new friend to the group) either by accessing his profile through a web access or by using an SMS with a predefined syntax (e.g. add "Carlo" to Myfriends). Figure 6 shows the service diagram for the latter option, In this case the SBB FamilyProfile1 is configured with the handler setProfile which will add the new user (defined in the SMS) to the group. The sendSMS block will finally send a confirmation to the user that the group has been modified.

7 Conclusions

The SCE developed by TILAB is a powerful tool to develop new services that integrate the capabilities available in fixed and mobile platforms. The tool fastens the whole creation process from the service idea to the actual service usage and it enables the service provider to deploy and modify several new services with low incremental investment (compared with full featured monolithic service platform). This approach is a strategic weapon for the operators, decreasing the risk of launching and customizing new services, in particular in the very difficult arena of the FMC.

References

G. Colombo et alii ,"Convergence trend for fixed mobile services".. IEEE Personal Communications. Vol 6, N 2 April1999.

G. Zaffiro , "Providing Wireless Broadband Services for Corporate Markets in a FMC Context".. IIR FMC conference , 2/2005.

JAIN SLEE specification Version 1.0, Java Community process, JSR 22, http://www.jcp.org/en/jsr/detail?id=22

JAIN SLEE specification Version 1.1 (under review), Java Community Process, JSR 240, http://www.jcp.org/en/jsr/detail?id=240

Falcarin, P., Licciardi, C.A., Analysis of NGN service creation technologies. In IEC Annual Review of communications, volume 56, June 2003.

Lago, P., Licciardi, C.A et alii., An architecture for IN-Internet hybrid services. In Computer Networks Journal, Special Issue on Intelligent Networks and Internet Convergence, T. Magedanz, H. Rudin, I. Akyildiz (Eds.), Elsevier, Vol. 35(5), April 2001, pp. 537-549

Andreetto, A., Licciardi, C.A., Falcarin, P., Service opportunities for Next Generation Networks, In Proceedings of the Eurescom Summit 2001, Heidelberg, Germany, November 2001.

Analysis of Movement Detection Process for IPv6 Mobile Nodes

Youn-Hee Han[1] and Seung-Hee Hwang[2]

[1] Communication & Network Lab.,
Samsung Advanced Institute of Technology, Yongin, Kyungki-do, Korea
`yh21.han@samsung.com`

[2] Department of Computer Science and Engineering, Korea University, Seoul, Korea
`shhwang@disys.korea.ac.kr`

Abstract. IETF Mobile IPv6 allows nodes to perform an IP handover without breaking their active sessions. In Mobile IPv6, after re-associating a new attachment of point, it is generally the responsibility of a mobile node to detect that it has moved to a new IP subnet. Determining whether or not a node has moved to a new IP subnet is not always a simple issue. In this paper, we analyze the movement detection process of IPv6 mobile nodes based on standards, i.e. IETF RFC documents and IETF Working Group Internet-drafts. We model the movement detection process and provide our results of performance analysis for the model in terms of latency caused by the process.

1 Introduction

In IP-based wireless and mobile network, a mobile node (MN) communicates with its correspondent nodes (CNs) via its IP address regardless of its location. Since the IP address is somewhat location-dependent, however, its IP address may be changed at its location change, and its communication also may be disconnected at its new location. To solve this problem, a lot of research has been studied so far on various fields: application layer like Session Initiation Protocol (SIP), transport layer such as mobile TCP, network layer like Mobile IP, and so on. The mobility support procedures in network layer are most popularly worked since they are directly involved to IP itself [1].

IETF Mobile IPv6 (MIPv6) [1,2,3] is one of the most important protocols to accommodate the increasing demand of end-to-end mobility in IPv6 Internet. MIPv6 enables an MN to change its point of attachment to the Internet whilst not breaking existing application sessions. When an MN is away from its home network, packets can still be routed to it using the MN's home address. Whenever an MN moves to a new subnet, it typically acquires its temporary address through the address auto-configuration [4] according to the methods of IPv6 Neighbor Discovery [5]. Through the binding update mechanism, an MN registers its temporal location to its home agent in its home network and all CNs every time it moves. After receiving a binding update message, CNs can directly send data packets to the MN without help of the MN's home agent.

MIPv6 handles problems related to MN's subnet change. At the time an MN establishes a new link-layer connection, a subnet change also might have occurred. In practice, it may not know whether its existing IP configuration is still valid for Internet connectivity. It also does not know whether its existing default router is on this link nor if its neighbor cache entries are valid. To decide these clearly, it is first required that the MN should rapidly check for the subnet change, i.e. determine whether a subnet change has occurred. This process is called *movement detection (MD).*

At the time an MN establishes a new link-layer connection, it may catch a *movement hint* which is an indication that a subnet change might have occurred. This hint initiates appropriate MD process. Movement hints come in various forms, and differ in how they indicate a possible subnet change. The period taken to receive a movement hint also varies.

The receipt of a movement hint is not necessarily a definite indication of having moved to a new subnet. Additionally, two decisions must be made: 1) a new and different access router is bi-directionally reachable, and 2) the previous access router is no longer reachable. These two decision processes are triggered immediately after receiving a movement hint. So, the total MD latency is composed by the period catching a movement hint and the maximum period between the completion times of two decision processes.

Real-time applications, such as VoIP and audio/video streaming, desire what is known as *seamless* handover, which indicates no (or very little) packet loss and low latency. Since it is usually known that MIPv6 itself cannot support the seamless handover, additional optimizations and/or changes to the protocol will be deemed necessary [6]. It is why IETF now makes effort to standardize new optimized mobility protocols for IPv6 nodes. It should be noted that the MD latency is a main constituent part of total IPv6 handover latency. It is the purpose of this paper to qualitatively and quantitatively analyze the current MD process based on standards: IETF RFC documents and IETF Working Group Internet-drafts [2,4,5,7]. We also provide an exact performance analysis in terms of latency caused by MD process. We believe that such a quantified analysis would be helpful to research in IPv6 mobility, especially IETF works targeting in movement optimization [7].

The remainder of this paper is organized as follows. Section 2 provides the current MD process in detail. Section 3 analyzes the performance of the MD process. Finally, our conclusions of this paper are in section 4.

2 Movement Detection Process

In this section, we describe the current MD process found in IETF standards.

2.1 Link-Layer Indication

The changes on the underlying link-layer status can be relayed to IP layer in the form of link-layer event notifications. The actual event is managed by the link-layer of an MN through execution of link-layer protocols and mechanisms. When the event successfully completes within the link-layer, its notification can be delivered to IP layer. By the time the notification is delivered, the link-layer of an MN must be ready to accept IP packets from its interface.

A basic link-layer event is considered potentially useful to MD process: link-up event. An MN may interpret *the link-up event notification* as a movement hint for a possible subnet change. With link-layer support, an MN can receive such a hint almost instantly [8].

For example, let us assume that an MN should establish an IEEE 802.11 link [9] with an access point (AP). In IEEE 802.11 network that supports Robust Secure Network (RSN) [10], successful completion of 4-way handshake between the MN and AP commences the availability of IP service. The link up event notification must be generated upon this event. In non-RSN-based networks, successful association or re-association events on the link-layer must cause a link up notification sent to the IP layer.

In this paper, the process to get a movement hint with the link-up event notification will be called H_{link}. In a node's protocol stack, the processing latency of a message or local trigger is ignored in this paper. Therefore, the period taken to receive a movement hint with link-layer support is just zero:

$$T_{H_{link}} = 0. \tag{1}$$

2.2 Network-Layer Indication

In the absence of such a notification from the link-layer, an MN should depend on an indication from network-layer where the IPv6 mobility protocol operates. An access router sends (pseudo-)periodic unsolicited router advertisement (RA) messages to its all nodes. Whenever such an RA is sent from an access router, a timer is reset to a uniformly-distributed random value between the router's configured MinRtrAdvInterval and MaxRtrAdvInterval, which are symbolized by R_m and R_M, respectively. When the timer is expired, new unsolicited RA is again sent. If the link-layer indication is not supported in the local protocol stack, an MN must interpret *the receipt a new RA from a new access router* as a movement hint.

There is another method to support network-layer movement hint. MIPv6 specification [2] defines a new RA interval option, which is used in RA messages to advertise the interval at which the access router sends unsolicited RAs. An MN can keep monitoring for periodic RAs and interpret *the absence of the exiting RAs* as a movement hint. The indicated interval is equal to the router's configured value, MaxRtrAdvInterval. To differentiate new access router's MaxRtrAdvInterval, the previous access router's MaxRtrAdvInterval will be symbolized by R_L. An MN may implement its own policy to determine the number of missing RAs needed to interpret that as a movement hint. In this paper, just one missing is assumed a movement hint.

In consequence, only with network-layer indication, a movement hint comes from two sources: 1) the receipt of new unsolicited RAs or 2) the notice of the existing RA's absence. In this paper, the process to get a movement hint only with help of network-layer will be called $H_{network}$. The average period $T_{H_{network}}$ to complete the process largely depends on the values of R_L, R_M, and R_m. We will mathematically analyze the period in Section 3.

2.3 Reachability of a New Access Router

First of all, it should be noted that the receipt of new unsolicited RAs must not be used as confirmation of bi-directional reachability to the new access router, which is stated in RFC 2461 [5]. They only confirm reachability in the router to MN direction.

In order to check if a new access router is bi-directionally reachable, therefore, an MN should receive a solicited RA after it sends Router Solicitation (RS) message to the all-router multicast address. This process is called *RC* in this paper. To commence *RC*, an MN should utilize one among the movement hints introduced in the previous subsection.

When exchanging RS and RA, according to RFC 2461 [5], it is sometimes necessary to delay a transmission for a random amount of time in order to prevent multiple nodes from transmitting at exactly the same time. The followings are two rules which should be kept when MNs and access routers exchange RS and RA messages.

- Before an MN sends an initial RS message, it should delay the transmission for a random amount of time between 0 and MAX_RTR_SOLICITATION _DELAY (=M_S) seconds.
- In all cases, RAs sent in response to a RS must be delayed by a random time between 0 and MAX_RA_DELAY_TIME (=M_R) seconds.

Assuming that message delivery time and its local processing time are ignored, therefore, the average period from getting a movement hint to complete *RC* is as follows:

$$T_{RC} = \frac{M_S + M_R}{2}. \tag{2}$$

2.4 Unreachability of the Previous Access Router

Due to the temporary packet flow disruption and signaling overhead involved in updating mobility bindings, MNs should avoid performing an L3 handover until it is strictly necessary. Therefore, an MN should decide if the previous access router is no longer reachable. To do it, an MN uses neighbor unreachability detection (NUD) procedure [2,5].

In NUD, each IPv6 node has a reachability state. The state transition between each states is described Fig. 1. When a node confirms that a neighbor is reachable, the state of that neighbor node is called *Reachable*. A node waits for REACHABLE_TIME ms before it changes the neighbor's state into *Stale*. During this state, nothing happens until the node sends a new packet to the neighbor. At the time the node sends a packet to the neighbor, the state of the

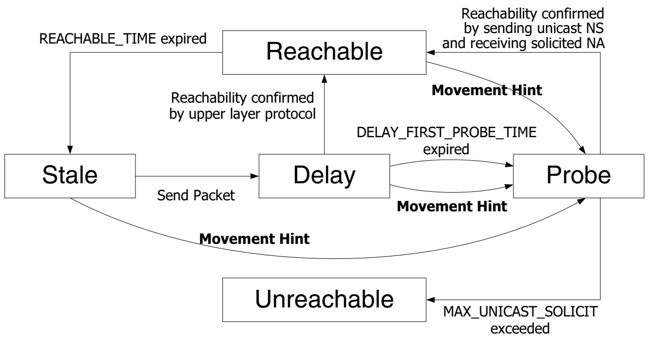

Reachable
Stale
Delay
Probe
Unreachable
REACHABLE_TIME expired
Reachability confirmed by sending unicast NS and receiving solicited NA
Movement Hint
Reachability confirmed by upper layer protocol
DELAY_FIRST_PROBE_TIME expired
Send Packet
Movement Hint
Movement Hint
MAX_UNICAST_SOLICIT exceeded

Fig. 1. The NUD process and its state transition

neighbor node goes to *Delay* and an actual NUD process begins. During this cycle, the node waits for another DELAY_FIRST_PROBE_TIME seconds and the state of the neighbor node goes to *Probe*. At this stage, the node sends Neighbor Solicitation (NS) messages to the neighbor to finally confirm the reachability with predefined number, MAX_UNICAST_SOLICIT ($=M_U$), of retransmission. The time interval between retransmissions of NSs is RETRANS_TIMER ($=T_R$). If the node does not receive any response from the neighbor during this cycle, the state of the neighbor node goes to *Unreachable*. Otherwise, it returns back to *Reachable*. These original NUD process is called NUD_o in this paper.

The period T_{NUD_o} to complete NUD_o becomes so large if the node has moved to a new network and the neighbor (the previous access router) is really unreachable. So, additional state transitions should be simply added to quickly finish NUD process. Fig. 1 shows the additional state transitions from *Reachable*, *Stale*, or *Delay* to *Probe*. Whenever a node catches a movement hint, the neighbor's state is allowed to immediately shift to *Probe* and the node sends NS to the neighbor. This new quick NUD process is called NUD_q in this paper. In fact, NUD_q is recommended in MIPv6 specification [2] and is reasonable for fast MD process. Thus the latency of this quick NUD process is as follows:

$$T_{NUD_q} = M_U \times T_R. \tag{3}$$

3 Performance Analysis of Movement Detection

In the previous sections, we described all stages to perform the MD process. In Section 2.2, however, we deferred the mathematical analysis of average period $T_{H_{network}}$ to complete the process $H_{network}$. In this section, therefore, we first analyze it and then derive the total latency of MD process.

3.1 The Average Period to Receive a New Unsolicited RA

The timing model in Fig. 2 illustrates the events of the arrival of new RA messages and link-layer re-establishment. T_n denotes the arrival times of the unsolicited nth RA come from new access router. To simplify notation, the last arrival time of an unsolicited RA from the new access router prior to a new link-layer connection is denoted by 0, and the first arrival time of such an RA after the new link-layer connection is denoted by R. The time an MN completes link-layer re-establishment is denoted by C. Let us assume T denotes the time interval between link-layer re-establishment and the first reception of a new RA. So, we have $T = R - C$. Because the time interval R is independent and uniformly distributed in the interval $[R_m, R_M]$, the distribution of C is independent of the interval in which it occurs.

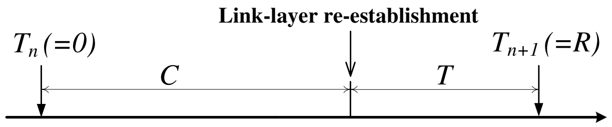


Fig. 2. Timing model of RA messages

Let $f_T(t)$ and $f_R(r)$ be the probability density function (pdf) of T and R, respectively. Because C is dependent of R, the joint pdf $f_{C,R}(c,r)$ can be given by

$$f_{C,R}(c,r) = f_{C|R}(c,r) \cdot f_R(r) \tag{4}$$

where $f_{C|R}(c,r)$ is the pdf for C given R and $f_R(r)$ is the pdf for the size of interval in which C occurs. Since C is evenly distributed in the interval $[0, R]$, we can calculate $f_{C|R}(c,r)$ as

$$f_{C|R}(c,r) = \frac{1}{r} \cdot 1_{c \in [0,r]} \tag{5}$$

where $1_{c \in [0,r]}$ is an indicator function with a value of 1 when $c \in [0,r]$ and 0 otherwise.

The pdf $f_R(r)$ expresses the probability that C should occur in the size of interval R. Intuitively, the probability of C occurring in an interval is proportional to the size of interval. When the size of interval is given as r, the function $f(r) = r \cdot 1_{r \in [R_m, R_M]}$ indicates this property. We can obtain the pdf $f_R(r)$ as $f(r)$ divided with the area of $f(r)$. this yields

$$f_R(r) = \frac{r}{\int_{R_m}^{R_M} r dr} \cdot 1_{r \in [R_m, R_M]} = \frac{2r}{R_M^2 - R_m^2} \cdot 1_{r \in [R_m, R_M]} \tag{6}$$

The joint pdf $f_{C,R}(c,r)$ is thus

$$\begin{aligned} f_{C,R}(c,r) &= f_{C|R}(c,r) \cdot f_R(r) \\ &= \frac{2}{R_M^2 - R_m^2} \cdot 1_{c \in [0,r]} \cdot 1_{r \in [R_m, R_M]} \end{aligned} \tag{7}$$

The pdf $f_T(t)$ can be obtained by integrating over the joint pdf $f_{C,R}(c, r)$ for all possible values of c and r. Since r can be expressed as $r = c + t$ (that is, $t = r - c$), the pdf $f_T(t)$ can be defined by

$$\begin{aligned} f_T(t) &= \int_0^{R_M} f_{C,R}(c, c+t)dc \\ &= \int_0^{R_M} \frac{2}{R_M^2 - R_m^2} \cdot 1_{c\in[0,c+t]} \cdot 1_{c+t\in[R_m,R_M]}dc \\ &= \frac{2}{R_M^2 - R_m^2} \cdot \int_0^{R_M} 1_{c\in[0,c+t]} \cdot 1_{c\in[R_m-t,R_M-t]}dc \\ &= \frac{2}{R_M^2 - R_m^2} \cdot 1_{t\in[0,R_M]} \int_{0\vee R_m-t}^{R_M-t} dc \\ &= \frac{2}{R_M^2 - R_m^2} \cdot 1_{t\in[0,R_M]} [c]_{0\vee R_m-t}^{R_M-t}. \end{aligned} \tag{8}$$

where $0 \vee R_m - t$ should be interpreted as the maximum value between 0 and $R_m - t$. From this equation, we then acquire

$$f_T(t) = \begin{cases} \dfrac{2(R_M - R_m)}{R_M^2 - R_m^2} = \dfrac{2}{R_M + R_m} & if\ 0 \le t \le R_m \\ \dfrac{2(R_M - t)}{R_M^2 - R_m^2} & if\ R_m \le t \le R_M \\ 0 & otherwise. \end{cases} \tag{9}$$

and its cumulative distribution function (cdf) $F_T(t)$ is as follows.

$$F_T(t) = \int f_T(t) = \begin{cases} \dfrac{2t}{R_M + R_m} & if\ 0 \le t \le R_m \\ \dfrac{2R_M t - t^2 - R_m^2}{R_M^2 - R_m^2} & if\ R_m \le t \le R_M \\ 1 & otherwise. \end{cases} \tag{10}$$

Having calculating the pdf $f_T(t)$, we can also obtain the average value E_T of t by

$$\begin{aligned} E_T &= \int_0^{R_M} t \cdot f_T(t)dt \\ &= \int_0^{R_m} t \cdot \frac{2}{R_M + R_m}dt + \int_{R_m}^{R_M} t \cdot \frac{2(R_M - t)}{R_M^2 - R_m^2}dt \\ &= \frac{R_M^3 - R_m^3}{3(R_M^2 - R_m^2)}. \end{aligned} \tag{11}$$

If we insert $R_m = 3$ sec. and $R_M = 10$ sec. in Equation (11), we obtain $E_T = 973/273 = 3.56$ sec.

3.2 The Average Period to Notice the Existing RA's Absence

Let us assume S denotes the time interval between the link-layer re-establishment and the notice of the absence of previous access router's RA. Recall that a new RA interval option indicates the interval at which an access router sends unsolicited RAs and the indicated interval of the previous access router is R_L. Therefore, if an MN keeps monitoring for periodic RAs and the existing RA does not come to MN until the time R_L elapses, an MN considers it as a movement hint. Because the time an MN leaves from the current subnet is independent of the time R_L, the pdf $f_S(s)$ and the average latency E_S of S are simply as follows:

$$f_S(s) = \frac{1}{R_L}. \qquad\qquad E_S = \frac{R_L}{2}. \tag{12}$$

3.3 Total Latency of MD Process

In order to get the average latency $T_{H_{network}}$ with the analysis of the above subsections, we first calculate the probability that an MN first gets a new unsolicited RA before it notices the existing RA's absence after link-layer re-establishment. The probability $P\{T < S\}$ is as follows:

$$\begin{aligned} P\{T < S\} &= \int_0^{R_L} P\{T < S | S = s\} f_S(s) ds \\ &= \int_0^{R_L} P\{T < s\} f_S(s) ds \\ &= \int_0^{R_L} F_T(s) f_S(s) ds = \frac{1}{R_L} \int_0^{R_L} F_T(s) ds \end{aligned} \tag{13}$$

Let us ignore message processing latency and message delivery latency in wireless/wired medium. We can acquire $T_{H_{network}}$ as follows:

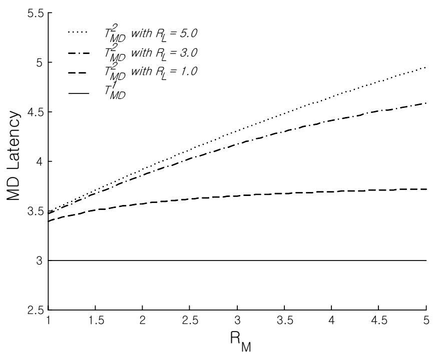
$$\begin{aligned} T_{H_{network}} &= P\{T < S\} \cdot E_T + (1 - P\{T < S\}) \cdot E_S \\ &= \int_0^{R_L} F_T(s) f_S(s) ds \cdot E_T + \left(1 - \int_0^{R_L} F_T(s) f_S(s) ds\right) \cdot E_S. \end{aligned} \tag{14}$$

From Equations (1), (2), (3) and (14), we finally get the average total MD latency. If an MN can utilize a link up notification, the notification can be an event to trigger the two test processes: RC and NUD_q. Therefore, the average total MD latency is as follows:

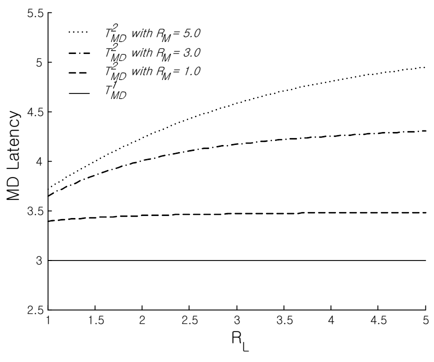
$$T_{MD}^1 = T_{H_{link}} + max\{T_{RC}, T_{NUD_q}\} = max\{T_{RC}, T_{NUD_q}\}. \tag{15}$$

On the other hand, when some MN cannot utilize such a link-layer notification, the two decision processes are triggered by the absence of RA from the default router or the receipt of new RA. So the average total MD latency in this case is as follows:

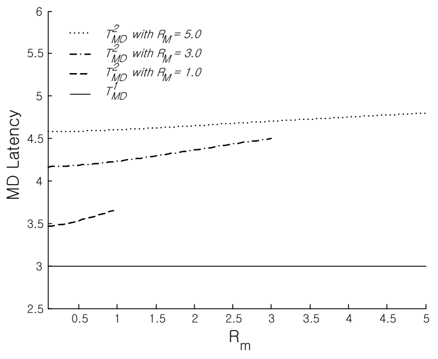
$$T_{MD}^2 = T_{H_{network}} + max\{T_{RC}, T_{NUD_q}\}. \tag{16}$$



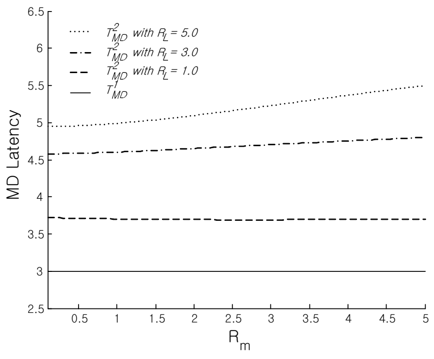
(a) $R_m = 0.1$



(b) $R_m = 0.1$



(c) $R_L = 3.0$



(d) $R_M = 5.0$

Fig. 3. MD latency comparison (unit: sec.)

With default values, $M_S = 1$, $M_R = 0.5$, $M_U = 3$, and $T_R = 1$, shown in [5], Fig. 3 describes the changes of T^1_{MD} and T^2_{MD} with regard to R_L, R_M, and R_m. As already known by Equations (15) and (16), T^1_{MD} is low and fixed value, while T^2_{MD} is high and usually gets higher as R_L, R_M, and R_m become high.

When a small value of 0.1 *sec.* is assumed for R_m, Fig. 3 (a) and (b) show that the values of T^2_{MD} increase steadily as R_M and R_L become high. However, it is noted that the increase gets slowed down. The reason for this is not difficult to grasp: without link-layer notification, an MN can get the movement hint from two notifications. Therefore, even though the period taken to acquire one notification becomes so long, the whole MD latency does not become long since the MN can obtain the other notification within its scheduled time.

With the fixed value of 3.0 *sec.* for R_L, Fig. 3 (c) presents that the effect of R_m on T^2_{MD} depends largely on the value of R_M. When R_M is low, R_m has much influence upon T^2_{MD}. In proportion as R_m rises, T^2_{MD} also increases. In contrast, when R_M is high, R_m is hard to make an effect on T^2_{MD}. The same case also occurs for diverse values of R_L and a high value of R_M, which is indicated by Fig. 3 (d). In the last

Fig. 3 (d), when R_L is very low ($R_L = 1.0$), T^2_{MD} rather slightly lessens when R_m increases. It is accounted for that $P\{T < S\}$ becomes small when both R_M and R_m are high and R_L is low. That is to say, E_S is relatively lower than E_T and the low E_S becomes the dominant factor to decide the value of T^2_{MD}.

4 Conclusions

This paper described the current MD process and showed a statistical approach in analyzing the performance of it in terms of latency. To the best of our knowledge, this letter does the first attempt to mathematically analyze the MD latency. As well as showing that it takes no time to get a movement hint with link-layer support, we also calculated a average period to get it only with network-layer support. The receipt of a movement hint is not necessarily a definite indication of actual layer 3 movement. Therefore, we also analyzed the period to decide two matters: 1) a new and different access router is bi-directionally reachable, and 2) the previous access router is no longer reachable. Finally, we presents how the total MD latency changes with regard to diverse values of R_L, R_M, and R_m.

Setting aside the variation of MD latency, it is clear that the current MD process is unnecessarily slow, so that it now hampers the use of current MIPv6 for real-time traffic. IETF DNA (Detecting Network Attachment) WG [7] is now active to define new standards to optimize the current MD process. We expect this paper would be a guide to exactly evaluate the MIPv6 performance, and be a good base document for achieving the goal of DNA WG.

References

1. Saha, D., Mukherjee, A., Misra, I.S., Chakraborty, M., Subhash, N.: Mobility Support in IP: A Survey of Related Protocols. IEEE Network **18** (2004) 34–40
2. Johnson, D.B., Perkins, C.E., Arkko, J.: Mobility Support in IPv6. IETF RFC 3775 (2004)
3. Vaughan-Nichols, S.J.: Mobile IPv6 and the Future of Wireless Internet Access. IEEE Computer **36** (2003) 18–22
4. Thomson, S., Narten, T.: IPv6 Stateless Address Autoconfiguration. IETF RFC 2462 (1998)
5. Narten, T., Nordmark, E., Simpson, W.: Neighbour Discovery for IP version 6. IETF RFC 2461 (1998)
6. Montavont, N., Noel, T.: Handover Management for Mobile Nodes in IPv6 Networks. IEEE Communications Magazine **40** (2002) 38–43
7. Choi, J., Daley, G.: Goals of Detecting Network Attachment in IPv6. IETF Internet draft, draft-ietf-dna-goals-04.txt (work in progress) (2004)
8. Yegin, A.: Link-layer Event Notifications for Detecting Network Attachments. IETF Internet draft, draft-ietf-dna-link-information-01.txt (work in progress) (2005)
9. Std. 802.11, I.: IEEE 802.11 Local and Metropolitan Area Networks: Wireless LAN Medium Acess Control (MAC) and Physical (PHY) Specifications. (1999)
10. 802.11i/d8, I.D.: Supplement to Part 11: wireless LAN Medium Access Control (MAC) and Physical layer (PHY) specifications: Specification for enhanced security. (2004)

A Dynamic Resource Allocation Scheme for Providing QoS in Packet-Switched Cellular Networks

Hermes Irineu Del Monego[1], Eliane Lucia Bodanese[2], Luiz Nacamura Jr[1], and Richard Demo Souza[1]

[1] LASD – CEFET/PR, – Curitiba-PR, 80-2390-901, Brazil
{hermes, nacamura, richard}@cpgei.cefetpr.br
[2] Dept. of Elec. Eng., Queen Mary University of London, London, United Kingdom
eliane.bodanese@elec.qmul.ac.uk

Abstract. In this paper, we present a dynamic bandwidth allocation strategy based on renegotiation. This strategy consists in exploring the unused resources in the network, allocating them to flows whose required bandwidth is greater than the one that was attributed to them at call admission time. These applications, which are non delay sensitive, can be admitted by the CAC (*Call Admission Control*) with the available bandwidth at the moment. Two scenarios are presented in order to show the functionality of the proposed system. The simulation results are analyzed and compared to the case of a system without bandwidth renegotiation.

1 Introduction

A lot of attention has been paid to the Quality of Service (QoS) provision in wireless networks, more specifically in the case of packet switched cellular networks such as GPRS, EDGE and UMTS [1]. Much of this effort has been made with the goal of providing end-to-end QoS, while prioritizing real-time traffic [2], [3].

In [4], the authors present a strategy for QoS renegotiation based on a priority-oriented call admission control for multimedia services over the Universal Mobile Telephone System (UMTS). In [5], a set of schedulers is proposed as a mean to guarantee a minimum bandwidth for each application. The unused bandwidth is then distributed among the active flows. In [6], the authors introduce a renegotiation system for Variable Bit Rate (VBR) traffic during a call execution. A resource allocation scheme is presented in [7], which guarantees QoS for real-time applications, without harm for the non real-time traffic. Similar mechanisms are also presented in [8-12]. Different CAC strategies are introduced in [13-17], with the objective of regulating the network traffic and providing QoS in a wireless network.

In this paper, we present a mechanism that allows bandwidth renegotiation for post-admitted calls. We utilize the model proposed in [16] for the CAC and scheduling. The mechanism prioritizes real-time traffic to the detriment of non delay sensitive traffic such as e-mail, ftp and www. Our renegotiation scheme makes use of resources unused by the higher priority applications and reallocates them to lower priority flows. The scheme also dynamically reallocates the bandwidth released after flow terminations. The reallocation of resources allows the traffic that was accepted with lower bandwidth than first requested to increase its allowed transmission rate.

This paper is organized as follows. In Section II we present the cellular network system model utilized in this work. The details of the proposed renegotiation scheme are presented in Section III. The implementation of the proposed system and the simulation results of two scenarios are shown in Sections IV and V, respectively. Section VI concludes the paper.

2 The System Model

The scheme proposed in this paper was specified in a GPRS/EDGE network. The General Packet Radio System (GPRS) is a packet switched cellular network developed over the GSM second generation cellular systems. In order to further increase the transmission rate over the radio interface, the Enhanced Data Rates for Global Evolution (EDGE) standard was introduced as a third generation communications alternative and it provided an evolution for GSM/GPRS systems [18]. In the GSM/GPRS and EDGE systems various mobile stations (MS), within a cell, are able to start/receive wireless communications through a base station (BS). The core of the GPRS/EDGE network utilizes the GPRS Support Nodes (GSN): 1. The GPRS Gateway Support Node (GGSN) acts as a logic interface for the external packet networks; 2.The Serving GPRS Support Node (SGSN) is responsible for the delivery of the packets to the MS through one or more base stations in a coverage area.

The base of the QoS management framework for GPRS was the introduction of the concept of the Packet Data Protocol (PDP) context. The PDP context is the logical connection set up between the MS and the GGSN to carry IP traffic [1]. A PDP context has a GPRS QoS profile defined in terms of the following QoS classes [2]: conversational, streaming, interactive and background. The QoS classes establish the fundamental bearer service characteristics. The conversational and streaming classes are associated to real-time traffic, and, therefore, they are delay sensitive. The interactive and background classes have looser delay requirements, however they are loss sensitive. One way to provide QoS in GPRS/EDGE networks consists in guaranteeing the desired requirements for the above classes.

In our system model, the CAC protocol, defined in [16], associates different priorities to the different QoS classes and allocates bandwidth differently according to the class of the requesting application. The CAC protocol aims at maximizing the number of flows per session or admitted calls over the wireless medium while keeping the QoS requirements.

Our admission scheme receives the addition of a dynamic feature that consists in renegotiating the bandwidth of the post-admitted calls within each class. The renegotiation is made based on the average bandwidth utilized by each data flow, and also based on the bandwidth released by any terminated flow.

Figure 1 shows the system model used in this work. A renegotiation function in the call management system is added to the basic structure of an EDGE network. The renegotiation function collects the information on the bandwidth usage in the MAC layer (explained in the next section), and renegotiates with the SGSN the bandwidth

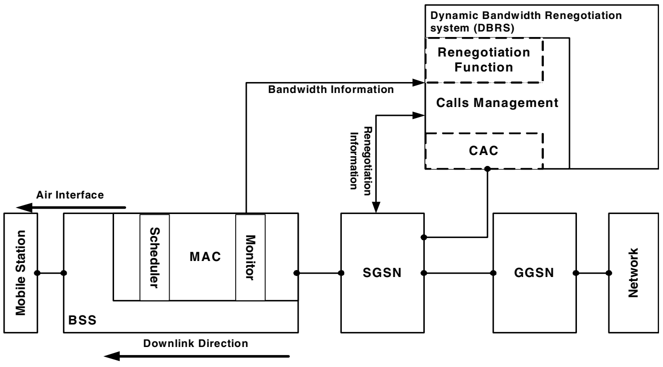


Fig. 1. The GPRS/EDGE network architecture and the proposed renegotiation mechanism

allocated to the active flows. The collection of this information at the MAC layer is done by a monitoring function. The collected data is, then, transferred to the renegotiation function.

3 The Renegotiation Scheme

In the CAC mechanism used in this work, conversational class applications are associated to a maximum priority (priority 1), and are admitted only if there is enough bandwidth at the request time. Priority 2 (intermediate) is given to streaming class applications, where again the requests are admitted only if there are enough resources. Priority 3 (the lowest priority within the mechanism) is associated with the interactive and background class applications. Priority 3 applications can be admitted with less bandwidth than the requested one. In the CAC defined in [16], the allocated bandwidth is kept constant even if more bandwidth becomes available in the system before the end of the admitted low priority application transmission. Another limitation of this CAC mechanism is that, if applications with priorities 1 and 2 do not effectively use the whole bandwidth allocated to them at call admission time, these unused resources can not be transferred to lower priority applications.

Our proposed renegotiation scheme has the objective of allowing that priority 3 applications use, temporarily, more bandwidth than the one allocated to them by the CAC. This possibility can be due to unused resources by applications with priorities 1 and 2, or due to the termination of another application of any priority. If an application with priority 1 or 2 arrives, and the system does not have enough bandwidth for admitting that call, the renegotiation mechanism can reduce the bandwidth being used by priority 3 applications to the value originally allocated to them by the CAC. This guarantees that applications with higher priorities will not be harmed by the renegotiation mechanism.

Therefore, the renegotiation mechanism consists in increasing the bandwidth of priority 3 applications when there are unused resources within the system, and to restore (decrease) the bandwidth of these applications at the arrival of an application with priorities 1 or 2. In the latter case, it occurs what we have called “renegotiation by priority demand”, while the former case, we have called “renegotiation by the average bandwidth and/or flow termination”.

3.1 Renegotiation by the Average Bandwidth

The renegotiation by the average of the utilized bandwidth consists in calculating the amount of unused bandwidth by the admitted calls. If the effectively used bandwidth is smaller than the admitted one, then the renegotiation starts and the unused resources are allocated to lower priority flows. Samples of the bandwidth utilized by the flows within the system are measured by the monitoring function. The quantity of bytes within each flow are summed during one time interval Δt. For each Δt, we obtain a partial average by dividing the number of transmitted bytes by the period Δt[1].

The n-th sample of the average used bandwidth can be calculated as:

$$\overline{Bm_n} = \frac{\sum_{p=1}^{P} Psize_{p_n}}{\Delta t_n}, \tag{1}$$

where, $Psize_p$ is the packet size, Δt is the duration of each sample and P is the number of packets. Thus, in order to obtain the average used bandwidth, $\overline{Bm_t}$; we have:

$$\overline{Bm_t} = \frac{\overline{Bm_1} + \overline{Bm_2} + \overline{Bm_3} + \cdots + \overline{Bm_N}}{N}, \tag{2}$$

where N is the number of samples.

Following the normal distribution, we can say that the average used bandwidth, $\overline{Bm_t}$, becomes reliable when the number of samples is larger than 30, $N > 30$ [19]. The standard deviation σ_b of the samples can be determined through the variance:

$$\sigma_b^{\,2} = \frac{\sum_{n=1}^{N} \left(\overline{Bm_n} - \overline{Bm_t}\right)^2}{N-1}. \tag{3}$$

As the standard deviation is calculated from the samples only and not from the whole population, we use the student's t-distribution [19] to approximate the values of the total used bandwidth within the interval:

$$\left[\overline{Bm_t} - t_{N-1}\frac{\sigma_p}{\sqrt{N}} ; \overline{Bm_t} + t_{N-1}\frac{\sigma_p}{\sqrt{N}}\right], \tag{4}$$

where, t_{N-1} is the constant of student for N-1 samples.

Then, as a conservative estimate, we use the upper limit of the above interval as the measured total used bandwidth Bt_m . In this case, we can determine the difference between the bandwidth admitted by the CAC (B_{wCac}) and the estimate of the total used bandwidth Bt_m :

$$B_\Delta = B_{wCac} - B_{tm} \tag{5}$$

where B_Δ corresponds to the unused bandwidth that can be renegotiated.

[1] Strictly speaking, we calculate the average data rates, not the bandwidth. However, in this paper we use the terms bandwidth and data rate interchangeably.

3.2 Renegotiation by Flow Termination

The renegotiation by flow termination consists in allocating more bandwidth for a low priority flow when another flow ends. The released bandwidth can be reallocated to another flow whose allocated bandwidth is smaller than the one requested to the CAC.

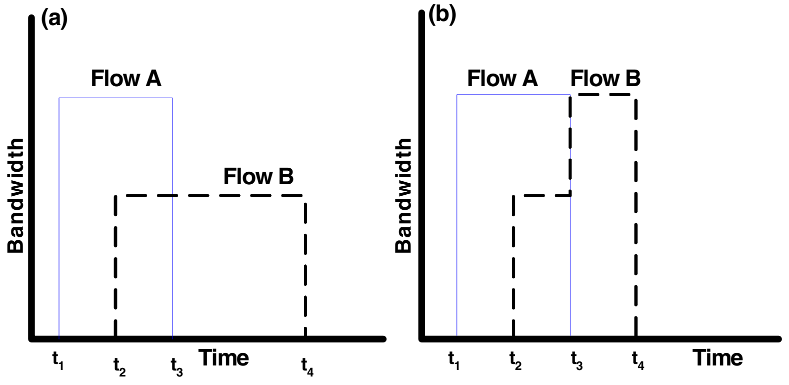


Fig. 2. Behavior of two different flows without (a) and with (b) renegotiation

Figure 2-(a) shows two different flows in a system without renegotiation. In this case, even though some bandwidth is available in the system after the termination of flow **A**, the bandwidth allocated to flow **B** does not change. Figure 2-(b) shows what happens in case of renegotiation by flow termination. Note that when flow A ends at time instant t_3, the renegotiation function increases the bandwidth allocated to flow B up to the requested amount.

4 Simulation of the Renegotiation Scheme

The renegotiation scheme was implemented in the NS-2 [20]. Two hypothetical scenarios were investigated with the objective of verifying the behavior of the data flows with and without renegotiation (by the average and by flow termination).

In the first scenario, we have generated data flows of ftp, voice, telnet and e-mail. The second scenario, more complex, contains data flows of video, music, e-mail, telnet and www. Table 1 presents the QoS classes associated with each application.

Table 1. QoS classes associated with each application under consideration

QoS Classes	Priority	Application
Conversational	1	telnet, voice
Streaming	2	music, video
Interactive	3	ftp, www
Background	3	e-mail

In the simulations, we have used the data flows available within NS-2 [20] for the case of telnet, ftp, music and voice applications. For the case of video and www applications we utilized the traces available in [21] and [22], respectively. Finally, for the e-mail we utilized the traces available in [23] and [24]. The average duration of each application was simulated according to [2], [9], [24], [25] and [26]. The number N of samples varied between 30 and 40 in order to satisfy the confidence constraints presented in Section 3.1-A. Table 2 presents a summary of the parameters used in the simulations.

Table 2. Parameters used in the simulations

Application	**Nominal Bandwidth (Kbps)**	**Average Call Duration (min-max)**	**Inter-arrival Time (s)**
Telnet	1.11	3 minutes (30s – max)	Exponential
Voice	4-25	3 minutes (60s – max)	Constant
Music	5-128	3 minutes (60s - max)	Constant
Video	20-384	6 minutes (100s – Max)	24 frames per second
Ftp	< 384	2 minutes (30s - max)	Exponential
E-mail	4.4	30 seconds (10s – 120s)	Exponential
www	-		Exponential

5 Simulation Results and Analysis

First we have defined the amount of bandwidth requested to the CAC by each application, which is 21.3 kbps for voice, 85 kbps for www, 1.11 kbps for telnet and 4.4 kbps for e-mail. As the ftp has priority 3, note that the allocated bandwidth can be smaller than the nominal value (85 kbps). In case of applications with priorities 1 or 2, such as the telnet that requires transmission rate of 1.1 kbps, the call can be admitted only if the full requested bandwidth is available.

Figure 3(a) shows the behavior of each data flow in the first simulation (with the CAC mechanism, but without the renegotiation scheme). The analysis of Figure 3(a) allows us to verify that, even though the applications ftp1 and ftp2 have requested the same bandwidth (85 kbps), ftp2 received a fraction of what was requested, because there were not enough resources within the system. The data flows corresponding to the applications with higher priorities (telnet and voice) were admitted with their nominal bandwidth values (1.11 and 21.3 kbps, respectively). The e-mail (priority 3) was admitted with its nominal value of 4.4 kbps, because at the request time there were enough resources within the system (note that the voice flow has already terminated). During the whole simulation, the mean bandwidth of each application practically did not vary.

Figure 3(b) shows the same scenario, but with the renegotiation scheme. In this case, the data flows of the active applications are monitored and the proposed mechanism can renegotiate any unused bandwidth either by priority demand, or by average or flow termination. Taking as reference the second ftp application (ftp2), we can see that at instant (1), the allocated bandwidth is already greater than the one allocated in Figure 3(a). At instant (2) the allocated bandwidth is increased due to the telnet flow termination. At instant (3), the renegotiation function is invoked again, due to the voice flow termination. Finally, after the termination of application ftp1, at time instant (4), the renegotiation function allocates the nominal bandwidth (85 kbps) to ftp2

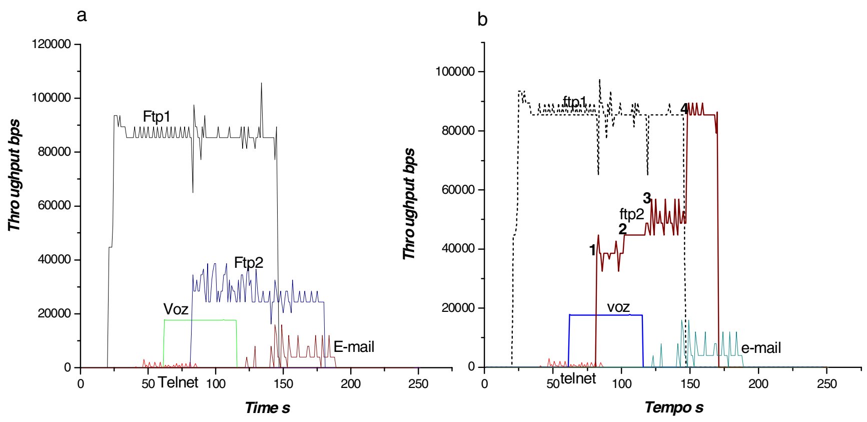


Fig. 3. Simulation of the first scenario with renegotiation

As we can see in Table 3, initially ftp2 requested 85 kbps, but the CAC allowed a transmission rate of only 38 kbps. The renegotiation scheme allowed a successive increase in the transmission rate of ftp2 to 39, 45, 58 and 85 kbps at stages (1), (2), (3), and (4), respectively. The ratio between the allocated and the requested bandwidth (85 kbps) increased from 44.7 to 100%. When the renegotiation scheme is compared to the simulation without renegotiation, the ftp2 average transmission rate, in Figure 3(a), is 32kbps and in Figure 3(b), it is 59 kbps. Therefore, there was a better usage of the available bandwidth with the renegotiation scheme.

Table 3. Numerical analysis of the ftp2 data flow

	Stage			
	1	**2**	**3**	**4**
Allocated bandwidth (kbps)	39	45	58	85
Ratio between the allocated and the requested bandwidth (85 kbps)	45.9%	52.9%	68.2%	100%

The second simulation scenario is composed of video, music, www, telnet and e-mail data flows. The amount of bandwidth requested to the CAC by each application is 65 kbps for video, 21.3 kbps for music, 65 kbps for www, 1.1 kbps for telnet and 4.4 kbps for e-mail.

Figure 4(a) shows the simulation results without renegotiation, while Figure 4(b) shows the results with the renegotiation scheme implemented in the system. Note that at stage (f) there is a decrease in the bandwidth allocated to the data flow corresponding to the www application. This happened because the music application was admitted into the system, and it has a higher priority than www. However, after the music flow terminates, part of the released bandwidth is recovered by the www flow (see stage (g)), while part of it is allocated to the e-mail flow that arrived during the execution of the music application. The increase and later decrease of the allocated bandwidth to the www flow indicates the occurrence of a renegotiation by priority demand.

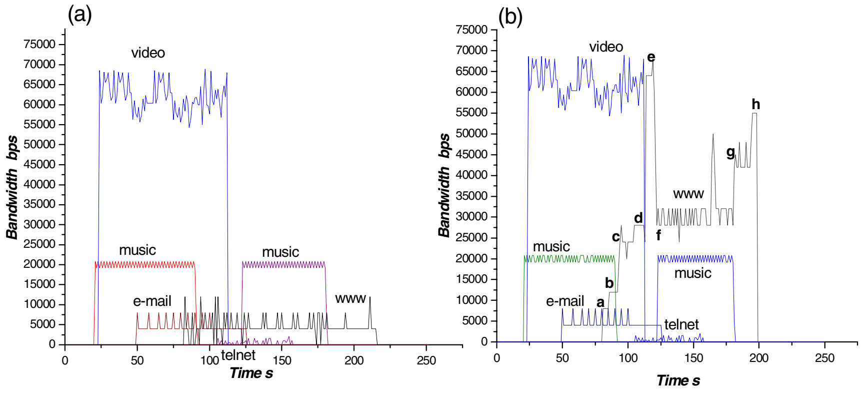


Fig. 4. Simulation of the second scenario with renegotiation

Taking as reference the www flow, Table 4 shows the variation of the allocated bandwidth during the www flow duration. Note that, at stage (e) the allocated bandwidth reached its nominal value, but after the admission of the music application the bandwidth was reduced to 45 kbps. As the originally allocated bandwidth is small (8 kbps), this result indicates that the renegotiation mechanism can be effective in high-demand systems where applications with low priority are initially accepted with a transmission rate much smaller than the nominal one. The graphs (a) and (b), in Figure 4, clearly show the increase of the allocated bandwidth for the www application in the system with the renegotiation scheme in comparison to the system without it.

Table 4. Numerical analysis of the www data flow

Stage	**Allocated bandwidth**	**Ratio between the allocated and the requested bandwidth (65 Kbps)**
a	8 Kbps	12.3 %
b	12 Kbps	18.46 %
c	24 Kbps	36.92 %
d	28 Kbps	43.07 %
e	65 Kbps	100 %
f	28 Kbps	43.07 %
g	45 Kbps	69.23 %
h	55 Kbps	84.61 %

7 Conclusions

This paper presented a renegotiation scheme that can be added to a GPRS/EDGE network. This mechanism aims at varying the bandwidth allocated to low priority applications, whose admitted bandwidth is usually smaller than the nominal one. In order to do not harm the higher priority applications, the scheme is also able to recover the renegotiated bandwidth and re-allocate it to higher priority applications at the time of their arrival at the CAC.

The analytical model of the renegotiation mechanism was implemented in the NS-2. Two hypothetical scenarios were presented in order to show the functionality of the proposed system. The simulation results were analyzed and compared with the case of a system without bandwidth renegotiation.

References

1. Halonen, T., Romero, J., Melero, J.: GSM, GPRS, and EDGE Performance Evolution Towards 3G/UMTS. John Wiley & Sons ltd Second Edition. England (2003)
2. 3GPP TS 23.107 UMTS Universal Mobile Telecommunications System, Quality of Service, v. 5.7.0, p.1-41. Available: www.etsi.org
3. Dixit, S., Guo, Y., Antoniou, Z.: Resource Management and Quality of Service in Third-Generation Wireless Networks. IEEE Com. Magazine, Vol. 39, No. 2 (2001) 125–133
4. Li, F.Y., Stol, N.. : A priority-oriented call admission control paradigm with QoS renegotiation for multimedia services in UMTS, IEEE VTC'01 Spring, Vol. 3, No. 53 (2001) 6-9
5. Yoshimura, T., Jeong, M.R., Imai, N., Morikawa, H., Aoyama, T.: Minimum bandwidth guaranteed scheduling for wireless and mobile network service, APCC/OEECC'99, Vol.1, No.5 (1999) 521–524
6. Reininger, D.J., Dipankar, R., Hui, J.Y.: Bandwidth renegotiation for VBR video over ATM networks, IEEE Journal on Selected Areas in Communications , Vol. 14, No. 6 (1996) 1076–1086
7. Armstrong, A., Hac, A.: A new resource allocation scheme for QoS provisioning in microcellular networks, IEEE ICC'98, Vol. 3 (1998) 1685–1689
8. Kim, S., Varshney, P.K.: An Adaptive Bandwidth Algorithm for QoS Sensitive Multimedia Cellular Networks, IEEE VTC'02 Fall (2002)
9. Oliveira, C., Kim, J.B., Suda, T.: An Adaptive Bandwidth Reservation Scheme for High Speed Multimedia Wireless Networks, IEEE Journal on Selected Areas in Communications, Vol.16, No. 6 (1998) 858-874
10. Xiang, C., Yuguang, F.: An adaptive bandwidth reservation scheme in multimedia wireless networks, IEEE GLOBECOM '03, Vol. 5 (2003) 2830–2834
11. Choi, S., Han, I., Ho Cho, D.: Resource allocation scheme for supporting high data-rate service in an integrated CDMA system, IEEE ICC'01, Vol. 6 (2001) 1733–1737
12. Jiang, J., Lai, T.: Bandwidth management providing guaranteed call dropping rates for multimedia mobile networks, IEEE ICME'00, Vol. 2 (2000) 963–966
13. Ho Chang, C., IL Myung, K., Sung, J. K.: Call admission control using the moving pattern of mobile user for mobile multimedia networks, IEEE LCN'02 (2002)
14. Jianping, J., Ten-Hwang, L.: Call admission control vs. bandwidth reservation: reducing handoff call dropping rate and providing bandwidth efficiency in mobile networks, IEEE ICPP'00 (2000) 21-24

15. Arregui, A., Dunlop, J.: Distributed call admission control in partially loaded TDMA systems, IEEE VTC' 98, Vol. 2 (1998) 1361–1365
16. Kochem, A.C.B., Bodanese, E. L.: A quality of service management scheme over GPRS, IEEE SoutheastCon 2003 (2003) 74–80
17. Yu, O., Khanvilkar, S.: Dynamic adaptive QoS provisioning over GPRS wireless mobile links, IEEE ICC'02, Vol. 2 (2002) 1100–1104
18. EDGE - Introduction of high-speed data in GSM/GPRS/networks: White Paper, available: www.ericsson.com.
19. Levine, D.M., Berenson, M.L., Stephan, D.: Statistics for Managers using Microsoft Excel, 2nd Ed, Prentice Hall Inc., USA (1999)
20. 20. The Network Simulator ns-2: Documentation. UC Berkeley available at: www.isi.edu/ nsnam/ns/ns-documentation.html
21. Trace Files: Available: http://trace.eas.asu.edu/TRACE/ltvt.htm.
22. Traffic Archive: Available: http://ita.ee.lbl.gov/html/contrib/ NASA-http.html.
23. Pang, Q., Bigloo, A., Leung, V.C.M., et al. : Service Scheduling for General Packet Radio Service Classes, IEEE WNC'99, Vol. 3 (1999) 1229-1233
24. Staehle, D., et al.: Source Traffic Modeling of Wireless Applications, Available: www3.informatik.uni-wuerzburg.de /TR/tr261.pdf
25. Traffic Model for IEEE 802.20 MBWA System Simulations: available in www.ieee.org
26. Staehle, D., et al.: QoS of Internet Access with GPRS, Research Report, Available: http://www3.informatik.uni-wuerzburg.de /TR/tr283.pdf (2002)

Distributed Authorization Framework for Mobile Agents*

G. Navarro[1], J.A. Ortega-Ruiz[2], J. Ametller[1], and S. Robles[1]

[1] Dept. of Information and Communications Engineering,
Universitat Autònoma de Barcelona,
08193 Bellaterra, Spain
{gnavarro, jametller, srobles}@ccd.uab.es
[2] Institute for Space Studies of Catalonia (IEEC),
80034 Barcelona, Spain
jao@gnu.org

Abstract. Mobile agent systems provide new perspectives for distributed e-commerce applications. These applications may present specific restrictions, making mobile agent systems a feasible solution. Even so, mobile agents present some security related problems. An important one is resource access control. The ability for mobile agents to provide a simple, scalable, flexible, and secure access control system is a key point for the widespread adoption of mobile agents. In this paper we propose a mechanism to safely assign roles to mobile agents and an access control method based on *Role-based Access Control* (RBAC). The access control method provides a simple, lightweight and distributed model for mobile agent applications. It is proposed as an extension of the MARISM-A (*An Architecture for Mobile Agents with Recursive Itineraries and Secure Migration*) project, a secure mobile agent platform.

Keywords: Access Control, Mobile Agents, Trust management, Security.

1 Introduction

During the last years, mobile agent technologies have witnessed an steady, if not fast, increase in popularity. Probably, the main hurdle to a wider adoption are the security issues that mobility brings to the picture [1]. Among them, an outstanding one is resource access control. Traditional access control methods rely on the use of centralized solutions based on the authentication of global identities (for example, via X.509 certificates). These methods allow to explicitly limit access to a given resource through attribute certificates or Access Control Lists, and rely on a centralized control via a single authority. Despite providing effective means of protection, these techniques suffer from serious drawbacks; in particular, they give raise to closed and hardly scalable systems. Practical mobile agent systems demand lightweight, flexible and scalable solutions for access control, in order to cope with the highly heterogeneous nature of their

* This work has been partially funded by the Spanish Ministry of Science and Technology (MCYT) though the project TIC2003-02041.

clients. In the same vein, solutions depending on centralized entities (such as traditional Certification Authorities) should be avoided.

There are alternatives based on *trust management*, which allow to assign authorizations (permissions or credentials) to concrete entities, as well as trust delegation among entities. Well-known implementations of these infrastructures are the *Simple Public Key Infrastructure/Simple Distributed Secure Infrastructure* (SPKI/SDSI)[2] and KeyNote[3], and several security frameworks are based upon it. Recent developments in this area, in an attempt to further ease access control management, have brought into the picture Role-based Access Control (RBAC) [4]. In these schemes, privileges of principals requesting access to a resource are determined by their membership to predefined roles. The use of RBAC greatly simplifies access control management and is specially suited to mobile agents scenarios, where agents privileges are subsumed in a possibly more general RBAC system.

This article presents an access control framework for mobile agents. We combine RBAC and *trust management* to provide a flexible, lightweight methodology for access control in such scenarios. In our approach, mobile agents do not carry any explicit information regarding resources access, avoiding the privacy concerns associated with sensitive data embedded in mobile code. In addition, our framework allows dynamical binding of authorizations to agents, providing thus great flexibility when it comes to define resource access control policies based on it. We have implemented our proposed scheme on MARISM-A, a JADE-based agent platform. Our framework is called DAFMA (*Distributed Authorization Framework for Mobile Agents*).

In Section 2 we introduce the main motivations behind DAFMA. Sections 3 and 4 introduce the main concepts and the base to our systems. We describe DAFMA and its components in Section 5. Finally, Section 6 summarizes our conclusions.

2 Motivations

DAFMA is intended to provide a suitable access control framework for distributed environments, and more specifically, mobile agent systems. There are not much proposals for access control systems for multiagent systems supporting agent mobility. Most of the security works in mobile agents deal with protecting communications, itineraries, and so on, but less of them deal with the protection of resources and the management of access control rights. Usually, proposed system rely on ad-hoc and centralized solutions. Just as an example, in [5] the authors propose a solution based on credentials, which are carried by the agent as authorizations. This credentials are combined with the Java protection domains to provide resource access control. The solution seem interesting but it relies too much on the platform itself, and specially Java. The system cannot be applied to other platforms and it is not distributed, since it requires a centralized policy.

In [6], the author presents a framework for building a secure infrastructure for agent systems based on SPKI/SDSI. Related to delegation it considers: chain-ruled delegation [7], threshold delegation, and conditional delegation. This approach does not consider mobile agent systems or even heterogeneous multiagent systems. The fact that agents are able to sign certificates with private keys, makes the whole system too much dependent in the trust relation between the agent and the execution platform.

JADE (`http://jade.tilab.it`) also provides a security add-on[8], which is also based on the Java security architecture and does not support agent mobility. FIPA (Foundation for Intelligent Physical Agents: `http://fipa.org`) also began to consider agent security through a technical committee, but the work has not been finished, and there is no FIPA recommendation at the moment on security related topics.

Other mobile agent platforms, such as NOMADS [9], provide access control based on Java-specific solutions and eventually used KAoS to provide high level policy languages and tools. KAoS [10] is a high level policy language currently focused on semantic web services. Despite its complexity, KAoS does not explicitly support mobile agents.

Our proposal provides a simple distributed architecture for the management of authorizations in mobile agent environments. Its ideas are, indeed, applicable to generic mobile code systems. A key factor is simplicity. We tried to simplify as much as possible the components and interactions, so the system becomes more easily implementable and scalable. DAFMA does not rely in implementation-dependent solutions such as the security architecture of Java, what makes it suitable for heterogeneous systems. And at the same time provides enough flexibility to accommodate a wide range of applications.

3 Distributed Naming Scheme

In DAFMA, we use a distributed naming scheme strongly influenced by SPKI/SDSI, which is used to create and manage roles and groups. We denote each entity (agent, platform, human user, etc.) as a *principal*, which has a pair of cryptographic keys (this is not exactly the case for mobile agents, see Section 5.5). The public key acts as a global identifier of the principal. In order to make it more manageable one can use the *hash* of the public key as an abbreviation for the public key. Each principal generates by itself the keys and is responsible of its own identity, so there is no need for a centralized CA, although it can be used if it is needed.

A principal can define local names of entities under its own responsibility. In order to do that, an entity has an associated local name space called *name container*, which can be made public to the rest of the world. The name container has entries of the type: `(<principal>,<local-name>)`. The *principal* corresponds to the principal for whom, the local name is being defined, and *local-name* is an arbitrary string. The *principal* can be specified as a public key or as a *fully qualified name* (see below).

For example, consider a principal with public key PK_0, which creates an agent with public key PK_1 and wants to name it *my-agent*. The name container of the entity will have an entry of the form: (PK_1, `my-agent`). Now on, the agent PK_1 can be referenced by the name *my-agent* in the local name space of PK_0. One of the most interesting issues is that a third party can make a reference to a name defined in other name containers through a *fully qualified name*. A name container is identified by the public key of the owner, so the fully qualified name PK_0 *my-agent* makes reference to the name *my-agent* defined in the name container of PK_0. It is important to note that given the properties of cryptographic keys, it is commonly assumed the uniqueness of the public key, so fully qualified names are globally unique.

These names, make it very easy for a principal to create groups or roles. For instances, a user PK_{admin} can create a group *employees* with members PK_a, PK_b and the agent PK_1, by adding the following entries in its name container:

$$(PK_1, \textit{employee})$$
$$(PK_2, \textit{employee})$$
$$(PK_0 \textit{ my-agent}, \textit{employee})$$

In order to support role hierarchies, we use group or role inclusion. That is, to declare a role as member of another role. For example consider the role *security-staff*, which is a super-role of *employees*. That is, members of *security-staff* also have the permissions or authorizations associated to *employees*. This could be expressed as:

$$(PK_{admin} \textit{ security-staff}, \textit{employee})$$

In order to make public an entry of a name container, we need to ensure that it cannot be forged. A *name certificate* is a name container entry signed by the owner of the container and including a validity specification. We denote such a certificate as:

$$\{(PK_{admin} \textit{ security-staff}, \textit{employee})\}_{PK_{admin}^{-1}}$$

where PK_{admin}^{-1} denotes the private key corresponding to the public key PK_{admin}, which digitally sings the certificate determining the issuer of the certificate or the owner of the name container where the name is defined. We do not show the validity specification for clarity reasons.

4 Authorization Certificates

The base to our proposal are *authorizations*. Authorizations are expressed through SPKI/SDSI *authorization certificates*. An authorization certificate has the following fields:

- Issuer (I): *principal* granting the authorization.
- Subject (S): *principal* receiving the authorization.
- Authorization tag (tag): specific authorization granted by the certificate.
- Delegation bit (p): if it is active, the subject may forward delegate the authorization received.
- Validity specification (V): validity of the certificate (time range and on-line tests).

The certificate is signed by the issuer. The on-line tests from the validity specification field, provide the possibility of checking, at verification time, the validity or revocation state of the certificate. For clarity reasons we will denote an authorization certificate as:

$$\{(K_s,\ p, +)\}_{K_i^{-1}}$$

Where the subject K_s received the authorization p from the issuer K_i. Again, K_i^{-1} denotes the private key associated with K_i, which signs the certificate. The plus symbol denotes that the delegation bit is active otherwise a single dash is used. Note that we represent an authorization certificate as a 3-tuple, while a name certificate is a 2-tuple.

Delegation introduces a lot of flexibility to the system. One principal can delegate authorizations to others without the intervention of a third party, authority, or without needing to modify any policy of the system. Despite its benefits, delegation comes with a cost. Suppose we have a bunch of name and authorization certificates and want to find if some principal is authorized to do something. Finding a certificate chain from a source of authority to the principal, that is, finding an *authorization proof* may become very complex, as it is in most sophisticated policy languages. In our case, we take as a base the SPKI/SDSI certificate chain discovery algorithm, which has a polynomial complexity and is easily implementable. It is based on simple reduction rules, such as:

$$\frac{\{(K_B, a, +)\}_{K_A^{-1}};\ b \preceq a}{\{(K_B, b, +)\}_{K_A^{-1}}};\ \frac{\{(K_B, a, +)\}_{K_A^{-1}};\ \{(K_C, b, +)\}_{K_B^{-1}}}{\{(K_C, a \wedge b, +)\}_{K_A^{-1}}}$$

Where '$\preceq$' denotes a partial order on authorizations (such as in lattice-based permissions), and '$\wedge$' denotes the intersection of authorizations. Previous to the reduction rules, there is a *name resolution* phase, which resolves names to public keys making use of name certificates. The reader may note that there is a little informality in this notation, for example the result of the first rule does not mean that there is a certificate signed by K_A granting permission b to K_B, but that this certificate (or its consequences) could be deduced from the previous statements.

5 DAFMA

The DAFMA system is implemented on top of MARISM-A, a secure mobile agent platform implemented in Java [11]. It provides extensions on top of the JADE system, a Java multiagent platform, which follows the standards proposed by FIPA. Mobility is achieved by the MARISM-A mobility component which is integrated into JADE. On top of JADE there are the main MARISM-A components such as the authorization framework presented in this paper, and other MARISM-A services such as: cryptographic service, directory service, service discovery, etc.

One of the first problems we found when planning the authorization model, is if a mobile agent should have a cryptographic key pair and be considered as a principal. A mobile agent cannot trivially store a private key, and cannot perform cryptographic operations such as digital signatures. There are some proposals to store sensitive information (private keys) in mobile agents [12]. But the problem arises when the mobile agent uses the private key to compute a cryptographic operation. The agency where the agent is in execution will be able to see the private key or at least, reproduce the operation. As a result we consider that a mobile agent should not have a private key.

Our solution is to establish the role membership of a mobile agent directly, in a way that the agent does not need to carry authorization related information, making the agent more simple and lightweight. This issue will be discussed in Section 5.5.

DAFMA is made up of four independent components, which interact to perform all the required functionality. This components are implemented as static agents. The interaction between this components is done using the FIPA ACL (*Agent Communication Language*) and FIPA ontologies. Figure 1 shows the components, which are described in the next sections.

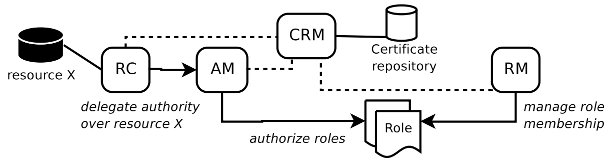


Fig. 1. DAFMA Architecture

5.1 Authorization Manager (AM)

The *Authorization Manager* (AM) manages the delegation of authorizations, issuing authorization certificates. It follows a *local authorization policy*, and its main responsibility is to delegate authorizations to specific roles following its local policy. It may also provide the ability to delegate authorizations to other AM in order to distribute the authorization management. Since the authorization policy is local to the AM agent, it does not need to follow any specification and its format can be implementation-dependent.

We propose an authorization policy, expressed as a sort of ACL (*Access Control List*) entry in S-expression format, similar to the SPKI/SDSI ACLs. An ACL entry is an authorization certificate without the issuer and it does not need to be signed because it is local to the AM and is not going to be transmitted. It has the following fields:

– Subject: the principal, who can request the AM to issue a certificate. Note that it does not need to be the principal receiving the authorization. This allows to separate the management of the authorization certificates (who makes the request) from the authorization itself (who receives the authorization).
– Authorization tag: the authorization tag matches the *authorization-request* issued by the principal asking for the certificate.
– Delegation bit: it indicates if the *Subject* can delegate the right to request the certificate.
– Validity: indicates the validity conditions (time range, ...) under which the certificate can be requested.

For example, suppose principal K_a wants to request to the authorization manager AM the certificate $\{(K_b, dummy, -)\}_{AM^{-1}}$, K_a will issue the following *authorization-request* S-expression:

```
(authorization-request
  (issuer AM)
  (subject $K_b$ )
  (tag dummy)
  (validity $V_1$ ))
```

Then AM could have the following SPKI ACL:

```
(acl
  ...
  (entry
    (subject $K_a$ )
    (tag (authorization-request ...))
    (validity $V_2$ )))
```

Note that the subject of the ACL entry and the certificate are not the same (a principal may request a certificate issued to another principal). The validity specifications V_1, and V_2 may also differ. V_1 indicates the validity for the request of the certificate and V_2 indicates the validity for the authorization granted by the certificate.

5.2 Role Manager (RM)

The *Role Manager* (RM) manages the roles (mainly role membership) by issuing name certificates following a *local role policy*. It can also assign a role to another role defined by itself or by another RM. Thus allowing the definition of role hierarchies or the delegation of role membership management. Section 5.5 details how roles are assigned to mobile agents.

Each RM has a local role policy, which determines what roles does it manage. It also includes rules to determine if a given principal requesting a role membership has to be granted or not. This is done by using a *membership-request*, which is equivalent to an *authorization-request*, and specifies the name certificate requested. If we choose to describe the role policy as a S-expression ACL, it is analogous to an authorization policy. The policy will determine which principal has authority to request memberships for a given role.

Now the subject of the SPKI ACL entry is a principal or another role, and the authorization tag determines the role that the subject can have.

An agent or principal, may belong to several roles. This is possible and adds lot of flexibility to the system. Since the system only uses positive authorizations or permissions, it is not possible to find conflicting authorizations for the same agent.

5.3 Resource Controller (RC)

The *Resource Controller* (RC) main task is to control the access to a resource. It holds the master SPKI key to access the resource, delegates authorizations to AMs, and verifies that an agent requesting access to the resource has a proper authorization.

It delegates authorizations to one or more AM following a local authorization policy. Note that this policy is quite simple because the main authorization management is performed by the AM. For example a RC controlling resource R_1 may delegate full control to AM_1 and read authorizations to AM_2.

5.4 Certificate Repository Manager (CRM)

The *Certificate Repository Manager* (CRM) implements and manages a certificate repository. For example, one agency may have one CRM to collect all the certificates issued by agents inside the agency. The CRM provides the repository and all the services needed to query, store or retrieve the certificates in the repository. It also provides a certificate chain discovery service. A principal can make a query to the CRM to find a specific certificate chain. This way we solve the problems derived from certificate distribution and leave the task to perform chain discoveries to the CRM and not to the other principals. It decreases communication traffic, certificates do not need to travel from one principal to another, and reduces the task that generic principals need to perform.

It is important to note that a distributed version of this module presents some problems mainly regarding with performance. Existing solutions [13] to provide distributed certificate chain discovery introduce limitations and complexity to the system. Another feasible approach is to use other mechanisms like a distributed LDAP directory, or use discovery services from the underlying agent platform [14].

5.5 Establishing Mobile Agents Role Membership

Since mobile agents cannot have private keys, we can not delegate authorizations to the mobile agent or make it member of a role. Our approach is to set, as member of the role, a hash of the agent's code. A principal can be identified by a a public key or a hash of a public key. So a hash may be seen as a principal, subject of a certificate. This is even supported by certificate frameworks, where a hash of an object can be considered as a principal, such as SPKI/SDSI, or even X.509.

In order to establish the role membership of a mobile agent we consider two different approaches. To show them we use a simple example where a user K_u is member of the role $physician$ defined by a Role Manager RM of the agency i, which has a resource controlled by a RC. The role $physician$ allows its members to access the given resource in agency i. The user has a mobile agent with the code m_i to be executed in platform i. The goal is to set the hash of m_i as member of the role $physician$.

User-managed role. The RM makes member of the role $physician$ a role (or group) defined by the user K_u, say $agent$. Then, the user K_u can make member of its role $agent$ any hash of agent's code.

$$\{(K_u \; \textit{agent}, \; \textit{physician})\}_{RM^{-1}}$$
$$\{(\textit{hash}(m_i), \; \textit{agent})\}_{K_u^{-1}}$$

RM-managed role. The RM makes member of the role $physician$ the user K_u. Then the users sends a request to the RM to set the agent code's hash as member of the role.

$$\{(K_u, \; \textit{physician})\}_{RM^{-1}}$$
$$\{(\textit{hash}(m_i), \; \textit{physician})\}_{RM^{-1}}$$

The *user-managed role*, is quite straightforward, gives the user full flexibility to manage the role K_u $agent$, and does not require any special mechanism or protocol. The user may add to the role any agent (or even user) she wants. The main problem with this approach is related to the accountability of the system. In e-commerce application there will be different degrees of trust between users. For example an hospital may trust an internal physician to manage its role $agent$, but a client from an external research center may not be so trusted. In the first case we will use *user-managed role*, while in the second one we will use *RM-managed roles*.

This last approach requires an additional protocol, which is implemented as an Agent Communication Language Interaction Protocol. It can be described in three steps:

1. The client sends an agent role assignment request (*ara-request*) to the role manager RM. Including m_i and the role she wants to obtain for the agent.
2. The RM requests the CRM to verify the client's role, and checks if he is member of the role requested in the *ara-request*.
3. If it succeeds, the RM computes the hash of m_i and issues a name certificate such as:

$$\{(hash(m_i),\ \textit{physician})\}_{RM^{-1}}$$

In the last step, the RM stores the code m_i, which may be used for further security audits. Note that this approach does not allow the user to manage the role and extend it to other agents. The user needs to send a request for each agent.

Later, the agent arrives to agency i, it will send an access request to the RC controlling the resource. The RC just has to compute the hash of the code m_i and check, through a request to the CRM, if the agent is member of a role authorized to access the resource.

The *RM-managed* approach can be used to directly authorize an agent. If the operation is carried out by an authorization manager it can delegate an authorization directly to the agent hash. The agent does not need to belong to a role. This does not introduce any conflict or problem to the system, since it is fully supported by SPKI/SDSI resolution and reduction algorithms.

The main drawback of this approach is that a mobile agent is no capable of issuing certificates, since the agent can not sign them. But note that this does not mean, that the agent cannot issue or delegate an authorization, which may be certified by a trustee.

Note that the authorizations associated to an agent may be determined by its role membership. This way we can say that the agent will have *dynamically assigned authorizations* during its lifetime. If the authorizations associated with a role change, the authorizations related to the agent also change.

6 Conclusions

We have proposed an access control system for a mobile agent platform. It provides a simple, flexible and scalable way of controlling the access to resources. It takes the ideas from RBAC and *trust management*. The proposed model is an extension of the MARISM-A project, a secure mobile agent platform based on JADE.

Currently we have implemented the main MARISM-A platform components (mobility, services, etc.), and a prototype of the authorization framework for access control proposed in this paper. Our solution provides a secure migration for agents with protected itineraries and we solve the secure resources access control and the authorization management. We provide a lightweight mechanism to authorize mobile agents, where the agent does not need to carry any kind of information regarding the access control. It introduces a method to establish and manage roles, which may be used in an open system with unknown users. This kind of users are normally untrusted and by the way authorizations are assigned, it will be feasible to introduce some accountability mechanisms to the system. The definition of access control policies also allows to separate the right of requesting an authorization from the authorization itself.

As a future work we plan to incorporate support for standard security languages, such as SAML to introduce more interoperability. Given the simplicity of the current framework this should not be a difficult issue.

References

1. Chess, D.: Security issues of mobile agents. In: Mobile Agents. Volume 1477 of LNCS., Springer-Verlag (1998)
2. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: RFC 2693: SPKI certificate theory. IETF (1999)
3. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.: The KeyNote Trust Management System. RFC 2704, IETF (1999)
4. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., Chandramouli, R.: Proposed NIST standard for role-based access control. In: ACM Transactions on Information and System Security. Volume 4. (2001)
5. Tripathi, A., Karnik, N.: Protected resource access for mobile agent-based distributed computing. In: Proceedings of the ICPP workshop on Wireless Networking and Mobile Computing. (1998)
6. Hu, Y.J.: Some thoughts on agent trust and delegation. In: Proceedings of the fifth International Conference on Autonomous Agents. (2001)
7. Aura, T.: Distributed access-rights management with delegation certificates. In: Secure Internet Programming: Security Issues for Distributed and Mobile Objects. Volume 1603 of LNCS. Springer-Verlag (1999)
8. JADE Board: Jade security guide. JADE-S Version 2 add-on (2005)
9. Suri, N., Bradshaw, J., Breedya, M., Groth, P., Hill, G., Jeffers, R., Mitrovich, T.: An overview of the nomads mobile agent system. In: Proceedings of 14th European Conference on Object-Oriented Programming. (2000)
10. Bradshaw, J. M., D.S.B.P..W.J.D.: KAoS: Toward an industrial-strength open agent architecture. Software Agents (1997)
11. Robles, S., Mir, J., Ametller, J., Borrell, J.: Implementation of Secure Architectures for Mobile Agents in MARISM-A. In: Fourth Int. Workshop on Mobile Agents for Telecommunication Applications. (2002)
12. Cartrysse, K., van der Lubbe, J.: Privacy in mobile agents. In: First IEEE Symposium on Multi-Agent Security and Survivability. (2004)
13. Li, N., Winsborough, W., Mitchell, J.: Distributed credential chain discovery in trust management. In: ACM Conference on Computer and Communications Security. (2001)
14. FIPA TC Ad Hoc: Fipa agent discovery service specification (2003)

Adaptive Content for the Mobile User: A Policy-Based Approach

Alvin Yew, Antonio Liotta, and Kun Yang

Department of Electronic Systems Engineering
University of Essex, Colchester, CO4 3SQ, UK
{kaayew, aliotta, kunyang}@essex.ac.uk

Abstract. Service adaptation is fundamental to context-aware service provisioning and to the realization of the Virtual Home Environment concept, which is now an integral part of the 3G service framework. Although the technological means for developing advanced, adaptable services are maturing, managing adaptation is still an ambitious task. In this paper, we tackle the issues of automating and managing adaptable web content for mobile users and illustrate the potential of policy-based service management. We present a novel policy-based framework, its experimental evaluation and discuss considerations on the applicability of state-of-the-art, XML-related technologies for the development of adaptable content and services for mobile users. We finally assess the overheads arising from the adaptation process and explore possible trade-offs.

1 Introduction

The wide variety of terminals that access the Internet has been increasing with advances in access network technologies. With increasing terminal and network access heterogeneity comes the challenge of finding suitable means, methods and technologies for a uniform, efficient service and content delivery. This challenge has partially been addressed by current technologies that allow the delivery of web-based content to relatively powerful terminals such as laptops and personal computers attached to broadband wired and wireless networks.

The situation is different for the case of content and service delivery to the mobile user with terminals of limited capabilities, which represents a rapidly growing slice of the telecommunications and Internet consumer markets. As the cost of owning Internet-enabled mobile terminals decreases, mobile users are likely to become the predominant user category. There is, therefore, a pressing need to provide new service delivery solutions that allow consumers to access a range of web-based content services whilst on-the-move regardless of terminal type and access network, thus delivering a homogeneous user experience without compromising service functionality. This may be particularly difficult from the content service provider's viewpoint since creating content specifically for every available terminal type may increase the complexity of managing and maintaining the actual content.

Many content service providers are looking at new, alternative approaches that will allow network operators and service/content providers to dynamically interact in order to provide on-the-fly, automated content adaptation. Automated service adaptation is also envisioned by the Virtual Home Environment (VHE), which is now an integral part of the 3G service framework [1].

In this paper we introduce a novel approach for delivering and managing automated web content adaptation, addressing the important requirements of terminal independence and ease of manageability. Our study includes the design and prototyping of a generic policy-based framework developed around XML-based technologies to provide adaptable web content to user terminals in 3G and beyond 3G (B3G) networks. We demonstrate the impact our approach can have on simplifying and speeding up the development of adaptable services through a prototype implementation and a content adaptation case study. We also present an empirical evaluation of the benefits of a policy-based approach –namely the added flexibility, ease of programmability and correctness– against the inevitable overheads associated with the system complexity. Research in policy-based management has been mainly associated with network and security management over the last few years; this paper illustrates that policy-based management (PBM) can also be beneficially applied to the management of adaptable services.

The paper is organized as follows. The next section presents background information by elaborating on the important technological building blocks for automated content adaptation, policy-based management for 3G/B3G networks, and defining content adaptation policies. We then proceed to give an overview of our framework, elaborating also on the arguments in favor of some crucial design choices. We subsequently discuss suitable strategies for defining policies for content adaptation to mobile terminals. The prototype evaluation and experimental results are presented in the penultimate section, and we conclude the article by elaborating on the implications of our results and the lessons learnt.

2 Background

Recent content provisioning frameworks based on XML technologies, such as the Apache Cocoon Project, represent the required evolution in adaptable web content provisioning and utilize the inherent flexibility of XML to make content reusable and easily adaptable by logically separating the actual ‘content’ from its ‘presentation’ [2]. Such frameworks could possibly use the recent W3C recommendation on Composite Capabilities/Preference Profiles (CC/PP) that provides terminals with an extensible and generic XML-based vocabulary to communicate their exact capabilities so as to allow web servers to make an informed decision when adapting the web content [3]. The CC/PP specifications have received growing endorsement in industry, resulting in the specification of a Java API for processing CC/PP, i.e. JSR 188 specification, by the Java Community Process [4]. Furthermore, the Open Mobile Alliance (OMA) has developed the User Agent Profile (UAProf) specification, which is a CC/PP–based vocabulary, specifically for mobile terminals to declare their capabilities [5].

XML-based frameworks and the CC/PP-related specifications together represent the fundamental building blocks of web content adaptation systems. However, a significant part of the adaptation process consists of the logic that decides how to best adapt the content in cases in which there are multiple possible solutions. This process can be interpreted as matching the content’s operating and accessibility requirement profile to that of the terminal capability profile [2]. [6] suggests using a rule-based approach consisting of simple 'if (condition) – then (action)' rules in a CC/PP-aware style-sheet to transform XML-formatted content into its final, adapted form through eXtensible Stylesheet Language Transformations (XSLT), i.e. transformations dependent on the values of CC/PP terminal capability attributes.

A rule-based approach is an appropriate solution to this process but performing the matching algorithm solely through a stylesheet has limitations in terms of flexibility because the intelligence of the adaptation is neither distributed nor partitioned. Although such an approach is still technically viable, it is not suitable for managing adaptation in the case in which different, conflicting conditions arise. On the other hand, a policy-based approach incorporates a well-defined information model as well as specific means for detecting and resolving conflicting conditions or rules [7,8].

2.1 Policy-Based Management in 3G/B3G Networks

Research in the principles of Policy-based Management (PBM) has reached an appropriate level of maturity over the last few years largely due to the efforts of the IETF's policy framework working group in standardizing various important aspects of PBM such as the Policy Core Information Model (PCIM) [8]. These well-defined IETF specifications were used as a reference guide by the 3GPP (3G's standardization body) when they included Policy-based Management functionality in its Open Service Access (OSA) interfaces, which is an important part of the VHE and 3G service framework [9]. The PBM related interfaces provide an avenue for future 3rd party service providers to perform PBM via an OSA gateway, thereby allowing advanced and flexible service management and control. It also provides content and service providers with a well-established framework in which to flexibly deploy, manage, and control adaptable services.

Figure 1 shows a scenario featuring the deployment of the OSA's PBM service capability by a network operator for the management of adaptable content for 3rd party content service providers. In this scenario, the network operator uses 3rd party content providers to offer a customized content portal as a VHE service for its customers. In release 6 of the OSA interfaces, the PBM functionality is represented by two different service capability features: Policy Provisioning and Policy Evaluation. The content service provider uses a policy management tool to interact with the Policy Provisioning Service Capability Server (PP-SCS) via the OSA gateway to define, create, modify, and delete policies for content adaptation in the policy repository housed in the network operator's domain. The Policy Evaluation Service Capability Server (PE-SCS) is used by the content provider to evaluate policies and generate policy events into the system so as to control the adaptation strategy and the provisioning of the content. The PE-SCS is particularly important when new policies are added, deleted, or modified in the policy repository as it is used to evaluate if there is a need to assert a change in policy for the current managed system. The content adaptation capability servers are responsible for interpreting and enforcing the policies when performing content adaptation, as well as retrieving policies from the policy repository when there is a change in the content adaptation and provisioning strategy.

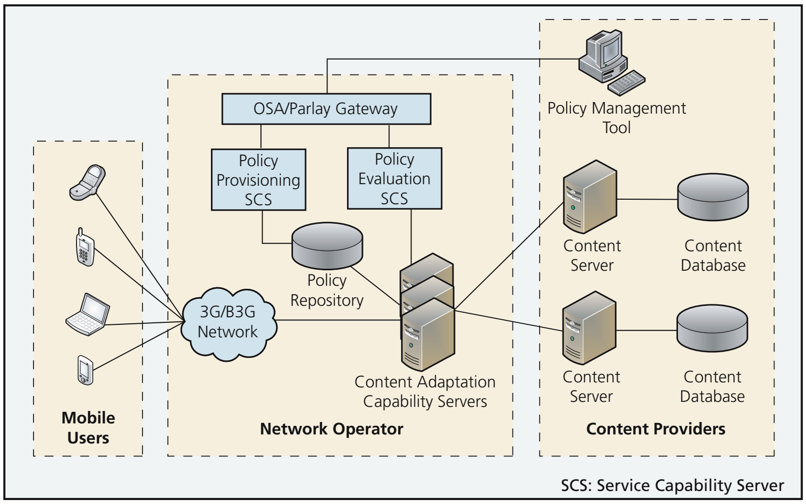


Fig. 1. Scenario deployment of policy based management in a 3G/B3G network

2.2 Declaring Adaptation Policies and Resolving Policy Conflicts

Policies are usually declared through numerous policy rules where within each rule a set of encountered conditions or events will result in the enforcement of a set of actions. The choice of policy conditions within a policy rule for delivering adapted content to mobile terminals can be any of the terminal capabilities declared in the UAProf specification. The resulting policy action is to deliver the content version that is best suited to the terminal. Policy conflicts generally occur in policy-based management when a given set of conditions may be applicable to at least two different policy rules with conflicting goals. In our framework, this would mean that there would be more than one version of a specific content suitable for the requesting mobile terminal, thus rendering the automated adaptation process ineffective. The example below shows a possible occurring set of conflicting conditions involving two separate policy rules for choosing different versions of a content-based service, and the example uses the policy notation specified in [8] (page 31).

```
Rule 1: if BitsPerPixel MATCH [2..32] && ImageCapable
MATCH true
Rule 2: if BitsPerPixel MATCH [2..8] && ColorCapable
MATCH true
```

The first rule is valid when the terminal display has a capability of displaying within the range of 2 to 32 bits per pixel and has also the ability to display images (in addition to text). The second rule is valid when the terminal display has a capability, in bits per pixel, comprised in the range of 2 to 8 bits and is also color capable. A policy conflict might therefore occur when the service is accessed by a terminal characterized by a resolution of 8 bits per pixel and capable of displaying color images. In this case both rules would evaluate to true.

Fortunately, the PCIMe defines two policy decision strategies that will solve such policy conflicts. The *FirstMatching* strategy involves placing a priority on a policy rule. The policy rules are then evaluated according to their priorities. On the contrary, the *AllMatching* strategy involves evaluating and enforcing the policy actions of all policy rules whose policy conditions are satisfied. The *AllMatching* strategy is not applicable to our framework and was therefore not considered.

At first glance, the *FirstMatching* strategy seems to be a solution for resolving adaptation policy conflicts in our framework. A UAProf device profile, however, need not declare every single device capability attribute available in the UAProf vocabulary, we encounter another problem evaluating policy conditions. This problem can be illustrated by using, again, the two rules defined previously and assuming that *Rule 1* has a higher priority than *Rule 2*. If a device profile does not contain any information about its ability to display images (refer to *Rule 1*), then there will be no way of evaluating if policy *Rule 1* is true, thus compromising the integrity of its policy-based decision. The solution to this problem is to assign a default condition evaluation outcome for policy *Rule 1* that the framework can fall back on when encountering such situations. The guidelines of the Common Information Model (CIM) state that it is acceptable to add properties to an existing CIM-derived class when extending a CIM-compliant model. Therefore, we added the new property *DefaultPolicyResolution* to the *PolicyCondition* abstract class in PCIM and PCIMe. The datatype of the value associated to this new property is a boolean attribute whose value represents the default outcome of evaluating the policy conditions. If a default policy rule is not defined when a policy conflict is encountered, then a fault is generated by our framework and delivered back to the terminal.

3 Description of the Policy-Based Framework for Delivering Adapted Content

3.1 Design Issues and Motivations

Our framework builds on the goals of the recent content provisioning frameworks based on XML technologies by incorporating the novel approach of policy-based management. The framework was designed from the ground-up, addressing in particular the requirement of separating web site management from content management. The framework's design was oriented to be XML-driven in order to pursue flexibility in delivering and provisioning content through the many XML-based specifications available for content formatting. For example, XML content can be transformed via XSLT to XHTML for web browsers, Wireless Markup Language (WML) for WAP browsers and devices, VoiceXML for voice-enabled web browsers and voice-driven content browsing. If the content is binary in nature (e.g. a video), however, then it does not undergo transformation by the framework.

Figure 2 shows a functional description of the proposed framework which is contained within the Content Adaptation Capability Server described in figure 1. The framework prototype is deployed using a Java Servlet based server – we used the Apache Tomcat Web Server – so as to utilize the Java CC/PP processing specifications. All the components within the Content Adaptation Capability Server in figure 2 have been developed from scratch with the exception of the XSLT 'markup transformation' component which is an Apache Xalan XSLT processor. The functionality and the roles of the various components in providing the adapted content service are further described in the following sections.

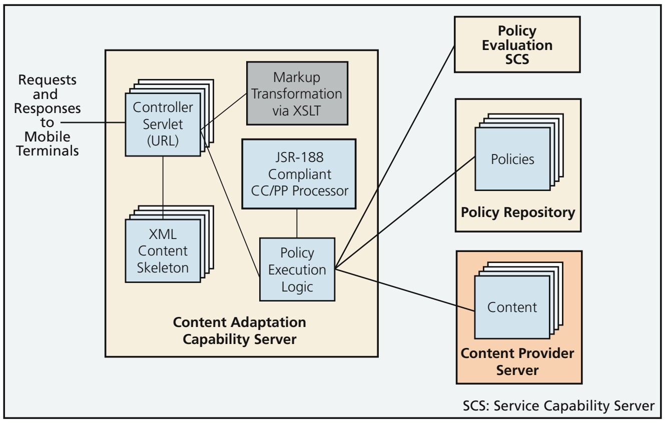

Requests and Responses to Mobile Terminals
Controller Servlet (URL)
Markup Transformation via XSLT
JSR-188 Compliant CC/PP Processor
XML Content Skeleton
Policy Execution Logic
Content Adaptation Capability Server
Policy Evaluation SCS
Policies
Policy Repository
Content
Content Provider Server
SCS: Service Capability Server

Fig. 2. The framework within the Content Adaptation Capability Server

3.2 The Controller Servlet

The Controller Servlet serves as the entry point for a web request for content from the mobile terminal to the framework. This request is performed through the Wireless Profiled HTTP (W-HTTP) or the CC/PP exchange protocols, as these are the two acceptable methods in which to convey UAProf or CC/PP information regarding the terminal's capabilities. The Controller Servlet is accessed by a consumer's terminal via the network by a Uniformed Request Locator (URL) and represents an accessible web resource within adaptable content service. Each Controller Servlet has an associated XML content skeleton. Figure 3 shows an example of a part of such a skeleton for a film review provided by an entertainment news service.

```
<Headline>Reviews of upcoming films</Headline>
<Review>
<AdaptComponent>
http://localhost:8080/films/SilenceOfTheLambs/
</AdaptComponent>
</Review>
<Picture>
<AdaptComponent>
http://localhost:8080/files/pics/SilenceOfTheLambs/
</AdaptComponent>
</Picture>
```

Fig. 3. Part of a XML content skeleton.

The content skeleton serves to declare the structure of the web content in a flexible and extensible manner through the use of XML. It may contain adaptable parts, which are represented by the presence of *AdaptComponent* tags, and non-adaptable parts. In figure 3, the *Picture* and the *Review* portions of the content service are adaptable while the *Headline* portion of the content remains constant.

The information within each set of *AdaptComponent* tags represents the location, which is defined as a URL, of the component responsible for the adaptation of the portion of the content skeleton in which the *AdaptComponent* tag is declared. The Controller Servlet's main functionality is to manage the adaptation process of the content as a whole by referring to the content skeleton. It leaves the adaptation of adaptable parts of the content skeleton to the component declared within the *AdaptComponent* tag, which is actually the Policy Execution Logic (PEL) component of the framework. For example, the PEL accessible at URL http://localhost:8080/films/SilenceOfTheLambs is responsible for adapting the *Review* portion of the content service in Figure 3.

The Controller Servlet reads the XML content skeleton when it is initialized and separates the adaptable part of the skeleton from the non-adaptable part after parsing the XML content skeleton. It does this so as to maintain a list of relevant Policy Execution Logic (PEL) components to invoke through the declared URLs when controlling the adaptation process. More importantly, in the Controller Servlet has to include information about the terminal's capabilities when invoking the PEL so as to allow the PEL to use this information when adapting the content.

Upon receiving the adapted content back from the PEL, the controller Servlet replaces the *AdaptComponent* tags portion in the XML skeleton with the adapted content. It does this, in turn, for every adaptable content portion of the XML skeleton until all adaptable content portions have been appropriately adapted. The final document is then transformed through XSLT for the appropriate markup language and before being relayed to the consumer's terminal.

```
<Headline>Reviews of upcoming films</Headline>
<Review>
Silence of the Lambs is a thriller about a imprisoned
cannibalistic psychopath who helps a detective solve
murders
</Review>
<Picture>
http://www.filmreviews.com/SilenceOfTheLambs/small.jpg
</Picture>
```

Fig. 4. Part of an adapted XML content skeleton

Figure 4 shows an adapted XML content skeleton after receiving content from the PEL in charge of adapting the *Review* portion of skeleton, which results in a short textual description of the movie. The other PEL in charge of adapting the *Picture* portion of the skeleton returned a link to a small version of the picture. The number of Controller Servlets varies in the framework as each Controller Servlet can serve more than one adaptable web resource.

3.3 Policy Execution Logic Component

The Policy Execution Logic component (PEL) is mainly responsible for adapting the adaptable portions of the XML content skeleton and delivering the adapted content back to the Controller Servlet. The PEL was implemented as a Java Servlet in our framework prototype and receives requests for adaptation from the Controller Servlet through the W-HTTP protocol. It is also possible to implement the PEL in other ways (e.g. distributed objects, Simple Object Access Protocol agents etc.) while also making the necessary amendments to the contents of the *AdaptComponent* tag of the XML skeleton (e.g. object reference for a distributed object implementation).

The PEL discovers information about the terminal capability by using the JSR-188 compliant processor (described further in Section 3.4) to construct an object representation of the terminal capability profile through using the UAProf or CC/PP information contained within the request.

Each PEL component maintains a reference to all the possible adapted content that it can deliver back to the controller Servlet. This may include textual, pictorial, and other multimedia content. The PEL must then decide which 'content' should be delivered on the basis of the terminal capabilities (obtained from the JSR-188 compliant processor) making a selection from a list of all the 'content' it is in charge of provisioning. The decision process is performed with the aid of adaptation policies that are stored in a repository. In our framework, policies are used solely for managing the adaptation process by controlling the PEL's content selection decision.

While each PEL Servlet represents a single adaptable web resource, a single policy can be made to be reusable for many PELs. This means that a group of PEL Servlets can possibly share the same adaptation policy and strategy. A programmer can thus input the location of the policy to the PEL Servlet's deployment description file, and the PEL will start 'consuming' the policy in the Servlet initialization phase. The policy associated with the PEL will then continue to govern its behavior for the period of its lifetime (i.e. until the Servlet is stopped and redeployed).

With respect to scalability, it is important to note that a PEL is associated with one adaptable resource, which itself can involve many static content formats and files used for the adaptation. Thus, we do not require a PEL for every single static media content file located in the content provider's server.

The final responsibility that the PEL has within the framework is that it is responsible for interacting with the Policy Evaluation SCS and the policy repository. These interactions inform the PEL of the creation, deletion, or modification of policies which would affect its current adaptation decisions. The PEL can then proceed to retrieve the new policy from the policy repository.

3.4 The JSR-188 Compliant Terminal Profile Processor

Before the specification of JSR-188, development of tools in industry for processing the CC/PP and UAProf vocabularies were done independently. As such, programmers often interpreted the specifications differently resulting in the situation whereby code that used a particular CC/PP processing API could not interoperate with that of other APIs. Consequently, evaluating the performance of CC/PP processing could not be done reliably as there was often no basis on which to make an objective evaluation. The development of the JSR-188 specification eradicates this problem as it provides a universal API that is independent of the underlying CC/PP processing logic for web developers to use. We developed a JSR-188 compliant processor that uses a Simple Api for Xml (SAX) parser to parse UAProf profiles. The main responsibility of the JSR-188 compliant processor in the framework is to provide CC/PP processing functionality to the Policy Execution Logic component in the framework. Only one processor is required to serve the whole framework.

3.5 Markup Transformation via XSLT

The Markup Transformation via XSLT component is used mainly to transform a document received from the controller Servlet into another (more appropriate) document and returns the transformed document back to the controller Servlet. The transformation is usually done to convert content into an appropriate markup for the consumer's terminal (e.g. XHTML, WML etc.).

4 Strategies for Defining Policies for Content Adaptation

There have been 6 different versions of the UAProf vocabulary since the year 2000 (to the time of publication), with a total of 91 definable attributes that are either locked, can be overridden, or appended to. Our experience with processing UAProf profiles has led us to wonder about the amount of attributes actually used by the average UAProf terminal profile. There are two significant benefits if a statistical analysis of UAProf profiles was available to policy and content authors. First, they could know what is the most often used UAProf capability declared by terminal manufacturers and can therefore plan the adaptable service around these capabilities. Second, the policy author can know the average number of attributes declared in a profile and can use that knowledge to plan the number of policy conditions to use in the adaptation. Knowledge of the structure and capability distribution of the average terminal can also be obtained from such statistics and can be useful for choosing a terminal profile for our experiments in a non-biased and informed manner. We conducted a survey of 98 different UAProf profiles available for download at [10]. Table 1 shows the results of our statistical analysis.

The total number of attributes that we scanned was 2892. Table 1 shows that the average UAProf profile has (rounded to the nearest whole number) 22 locked attributes, 5 Append attributes, and 3 override attributes, making it a total of 30 terminal capability attributes declared. Using the 98 different UAProf profiles that we surveyed, we also obtain a list highlighting the popularity of specific UAProf attributes. The content adaptation strategist can therefore plan the conditions for content adaptation using this list. Figure 5 shows the ranking of the top 25 UAProf attributes (according to the number of occurrences) that appeared in our statistical study.

By referring to the statistical data on the distribution of attribute datatypes, we decided to use the Sony-Ericsson T200A UAProf profile as the choice of terminal profile when evaluating our framework as its UAProf profile has the closest match to the average UAProf profile in table 1.

Table 1. Statistical analysis of UAProf profiles

	Locked	**Append**	**Override**
Occurrence	2117	527	248
Percentage of Occurrence	73.2	18.2	8.6
Average per profile	21.60	5.38	2.53
Median (per profile)	21	5.5	2.5
Maximum Possible attributes (per profile)	60	18	13

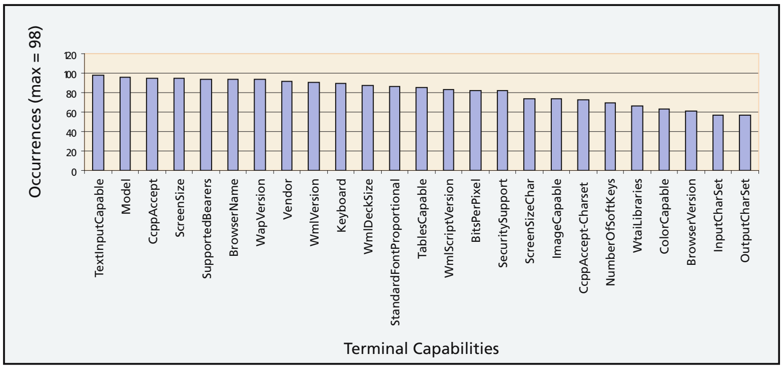


Fig. 5. Frequency of occurrences for specific CC/PP capabilities

5 Experimental Results and Analysis

We assessed the performance of our framework through a local area network testbed, ensuring that network delay had negligible influence on the results. Our server had an Intel Pentium 4, 2.53 GHz processor with 512Mbytes of memory. Apache Tomcat version 4.1.27 was used to support the framework, with a Java 1.4.1 Virtual Machine. The server was deployed using the 'server' option of the Java virtual machine to improve performance. Our experiments were planned as follows: first, in order to assess the performance of a non-policy based version of the system we examined the effects of using just Java Servlets with our CC/PP processor for the delivery of adaptable web content. We then repeated our experiments with a policy-enabled system, gradually increasing the number of adaptable components (i.e. *AdaptComponent* in the XML skeleton) from 0 to 4. Throughout the experiments, we used the same policy consisting of two rules, each of which contained a simple policy condition for each adaptable component. We created experimental conditions that were CPU-bound rather than disk-bound by constructing relatively simple adaptable web content. We used Apache JMeter (version 1.9.1) to measure the response times and throughput, and the results were obtained through measuring 2000 web requests.

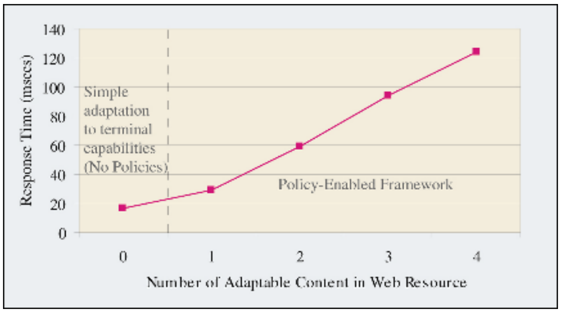


Fig. 6. Average response times of experiments

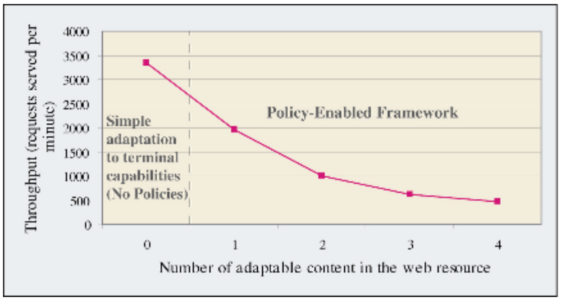


Fig. 7. Framework throughput performance

Figure 6 shows the average response times of our framework as observed by JMeter, while Figure 7 shows the throughput of the server in terms of requests served per minute. In both diagrams we observe a change in slope which corresponds to the activation of the policy-based engine, generating an increase in the performance degradation rate. The response times increases in a linear fashion, while the throughput of the framework degrades gracefully. These results reflect the cost paid for the additional flexibility, programmability and manageability offered by the policy-based approach.

6 Conclusions

As the number of terminal capabilities increases in the future, the options for flexible service provisioning become greater and adaptable service provisioning represent an increasingly important requirement for consumers in the future. It is fairly straight-forward to expect a loss in server performance when providing such services, but a scientific and empirical approach to quantifying this loss is certainly needed. This paper explored the applicability of a policy-based framework to provision adaptable services, highlighting the added flexibility in managing the adaptation process. We described how policies can be modeled, defined and interpreted to control the adaptation, as well as the methods that can be used to avoid policy conflicts. Second, we described guidelines on what terminal attributes to adapt to and how much adaptation is necessary by analyzing statistically the popularity of specific device attributes, and the number of device attributes declared in terminal profiles. Finally, we attempted to quantify the loss in server performance against the adaptability of a web content service through experiments with our prototype framework. Though the experiments do show a loss in server performance, this loss can be easily encountered in the feature by more powerful servers, and techniques such as load-balancing. In this respect, policy-based service management is certainly a promising approach toward increasing the level of automation in the deployment of advanced services.

Acknowledgements

The technical work presented in this paper has been developed in the context of the POLYMICS project, funded by the UK Engineering and Physical Sciences Research Council (EPSRC) - Grant GR/S09371/01.

References

1. 3rd Generation Partnership Project Technical Specification TS 23.127, Virtual Home Environment (VHE) / Open Service Access (OSA); Stage 2.
2. A. Pashtan, S. Kollipara, M. Pearce: Adapting Content for Wireless Web Services, IEEE Internet Computing, Sep/Oct. 2003, Vol. 7, No. 5, pp. 79-85.
3. G. Klyne et al (eds.): Composite Capabilities/Preference Profiles (CC/PP): Structure and Vocabularies 1.0, W3C Recommendation dated 15 January 2004, available at "http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/".
4. Luu Tran, Mark Butler (eds.): Composite Capabilities/Preference Profile (CC/PP) processing specification, version 1.0 dated 11 Jun. 2003, Sun Microsystems Java Specification Request 188.
5. Open Mobile Alliance: User Agent Profile 1.1, Candidate Version dated 12 December 2002, OMA-WAP-UAProf-v1_1-20021212-c.
6. Mark Butler: Using capability classes to classify and match CC/PP and UAProf profiles. HP Labs Technical Report HPL-2002-89 dated18 Feb 2002.
7. A. Westerinen et al.: Terminology for policy-based management, IETF RFC 3198, November 2001.
8. A. Moore (ed.): Policy Core Information Model (PCIM) extensions, IETF RFC 3460, Jan. 2003.
9. 3rd Generation Partnership Project Technical Specification TS 29.198-13, Open Service Access (OSA); Application Programming Interface (API); Part 13: Policy management (Release 6)
10. w3development website: "http://w3development.de/rdf/uaprof_repository/"

An Authorisation and Privacy Framework for Context-Aware Networks

Raffaele Giaffreda[1], Hannes Tschofenig[2], Theo Kanter[3], and Christoph Reichert[4]

[1] British Telecommunications plc, United Kingdom
raffaele.giaffreda@bt.com
[2] Siemens, Germany
hannes.tschofenig@siemens.com
[3] Ericsson, Sweden
theo.kanter@ericsson.com
[4] Fraunhofer FOKUS, Germany
reichert@fokus.fraunhofer.de

Abstract. Making network-centric context information available to hosts, network nodes and users allows them to make more meaningful decisions to accomplishing a self-organising behaviour. Instead of implicitly offering services as today we suggest a shift towards explicitly expressing context information that can be used with user context information to allow applications to operate more efficiently . Since security is of great concern when context information is accumulated and made available we combine it with a privacy-sensitive authorisation framework. This security framework is flexible and powerful enough to operate on both user- and network-context information. The paper demonstrates how a context-aware network architecture can be mapped to the IETF GEOPRIV privacy and authorisation framework to restrict and control the distribution of context information.

1 Introduction

One recurring issue when dealing with systems that claim to implement some sort of context-awareness, concerns the distribution of data needed to achieve automatic adaptability of systems' behaviour to context. Certainly lack of privacy or, to be more specific, lack of control over who can use and distribute a particular piece of context information may hinder the availability of such info in general, reducing the efficacy of any context-aware system. In this paper we look at the architecture that has been designed to support the operations of an Ambient Network with context-awareness and we propose a framework based on already existing IETF work to provide access control functionality.

The paper is organised as follows. Section 2 summarises the background information concerning context aware networks and how these can be related to the user context. Section 3 introduces the privacy requirements of the architecture for context management and positions related work. Sections 4 and 5 suggest a framework for securing access control to context information based on the IETF GEOPRIV privacy framework. Section 6 wraps up the paper with concluding remarks and indications for further work.

2 User Context and Context Aware Networks - Background Information

The concept of context-aware computing [10] has been around for a number of years now and has been implemented in many disguises, mainly to improve the operations of applications at mobile nodes. In spite of its longevity as a research topic, the use of context-awareness still has unexplored potentials if applied "few layers below", namely exploiting network context to improve networking applications. These potentials and the challenges documented in [5], are partly addressed by the Ambient Networks project [2], which has the aim of providing a "zero-configuration" network architecture taking, among many other building blocks, benefit of context information.

A detailed description of such architecture is out of the scope of this paper (interested readers can find more information in [8]), therefore we will only briefly introduce here terminology and building blocks that are used in the remainder of this paper. ContextWare is a component of the Ambient Control Space [7] designed for context information provisioning in an Ambient Network (AN). It includes two functional areas, the Context Coordinator (ConCoord) for supervising access to and the Context Manager (CM) for managing the content of a distributed Context Information Base (CIB).

The CIB, a distributed logical repository, is used to access, process and distribute mainly, but not exclusively network context. "Network context" can be a number of specific information that characterise an Ambient Network. It could be cost of network access, location, roaming relationships provided by a network, QoS model and QoS signalling support, presence of middleboxes (such as NATs and various types of firewalls), privacy statements and privacy policies, availability of Internet access, current network performance status (e.g., network load) and available facilities (e.g., a list of service descriptions available at the network, shared hardware devices available to its users etc.). Services might be identified by attributes, functionality provided according to a certain RFC, a registered service identifier or a service description using the WSDL/UDDI framework [15].

Although the ContextWare architecture focuses on network context, the CIB is not totally unrelated from the user's context. Elements of the user's context, which do not apply to the ontologies available in the Ambient Network control space, cannot be a source for reasoning about context within Ambient Networks. For example, decisions about pre-caching of data by a proxy located in the network to ensure seamless operation in a delay tolerant network (such as a train that passes a tunnel or a bus that only periodically offers connectivity) can be made more efficient by associating context information of the user with the context of the network. We believe that there are cases where user specific information can be exploited as a context source in the ContextWare architecture and vice versa. The user information includes presence status, location information, privacy preferences, etc.

A mechanism to associate or correlate the application layer identity of a user with the users network identity and also associate context data with this identity is further discussed below and illustrated in Figure 1. The picture shows the example of user specific presence information [1] being extended with other user context information and the potential relationship with the ContextWare framework. A Context Provider Agent (CPA) [6] is an example of a possible means for aggregating context relevant to a particular user and notifying the ConCoord of the availability of this info, which logically makes it become part of the CIB. These user related attributes can then be associated with the network context and build complete context snapshots that can be used to influence the way services are delivered through various networks to the user.

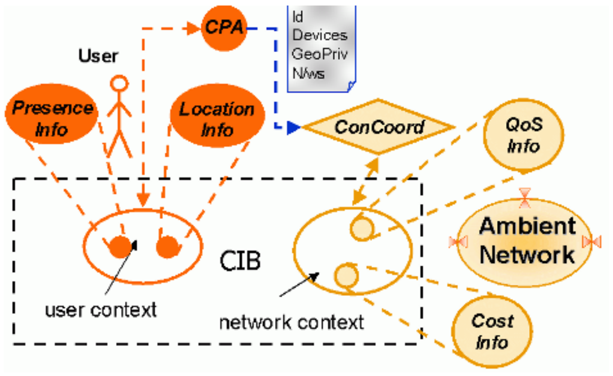


Fig. 1. User Context versus Network Context

The potential links between the ContextWare architecture and systems for managing user context have been introduced to make the need access control to the context information and to address privacy concerns more obvious. The remainder of the paper will not make the distinction between user and network related context as we believe that the validity of the suggested authorisation and privacy framework also applies to protect user specific privacy. This is also in the spirit of the Ambient Networks project where every user can potentially operate a network and privacy for the network owner corresponds to privacy of a user.

This paper's main contribution is on authorisation and privacy, with the term *privacy* referring as "control of information" as in [11]). This document does not address the aspect of guaranteeing that the sources of context information are trustworthy, that the provided context information is genuine and that these sources are allowed to generate specific context information (i.e., providing ownership of the distributed context information). This aspect has been recognised as an important aspect of deployable ContextWare architecture in order to avoid malicious nodes injecting deceptive context information in order to destabilise the operation of a network and its end host if context information is used to trigger certain actions.

3 ContextWare and Its Privacy Requirements

In the previous section we briefly illustrated the main characteristics of the ContextWare architecture. The main role of this architecture within an Ambient Network is to first provide a distributed storage and second to reason about an entity's context information, where an entity can be not only a user but also a network device or even an entire network.

Existing work, such as the Context Fabric [4], focuses on the privacy of end users with regard to presence specific information, e.g. status (busy, idle, available), location information, etc. These academic approaches are very closely related to the IETF work on SIP presence [12] and tend to provide application layer solutions only.

While reasoning about context in a network infrastructure is a novel idea, distributing and advertising information in a network (referred as network context information) can already be achieved with a number of protocols such as DNS, DHCP, SNMP, ICMP Router Advertisements, specialised discovery protocols, such as SLP, UPnP or Salutation, etc. Various security threats have been investigated for these protocols and their countermeasures can partially be reused. In this section we discuss how the ContextWare architecture can be positioned with regards to those existing solutions, justifying the new approach taken.

The term "context" is used to describe the circumstances in which events occur. Those *circumstances* are usually described with data that can be of a large variety of types. Context information by its nature can originate from a large number of sources. In certain situations it can be sufficient to use a directory lookup service (e.g., using the Domain Name Service), in other situations context information can be automatically discovered (e.g. by listening to link layer beacons, Router Advertisements or information distributed by a DHCP server). Yet other scenarios require protocols that support a subscription-alike semantic whereby performance constraint clients might expect information reduction (via rate limiting, transformation or aggregation) to be offered by network elements before a response is generated. Also, the language used to describe context information must be semantically richer than what can be achieved with many current protocols for storing and distributing data in the network.

The type of context information, the higher update rate, the subscribe/notify style of the information delivery leads to a restricted applicability of existing protocols since they are often specialised for a particular environment (e.g., distribute information only within a subnetwork) or task (e.g., distribution of infrequently changing information). The ContextWare architecture was designed to provide a high degree of flexibility for wide range of services. A framework for access control must take this environment into consideration. When privacy and access control mechanisms need to be provided, one must consider the fact that context information can be either stored directly at its source and referenced by the CIB, it can be copied at the CIB, it could be created within the CIB or it can be re-distributed. These aspects affect the way policies are managed as it will be illustrated in the next section.

To achieve proper privacy protection and authorisation for access to context information it is necessary to ensure that the following goals are met:

– Usage of authorisation policies as a mechanism to restrict distribution. Authorisation policies are a convenient way to express the context provider's intent regarding the distribution of context information to other entities. The policies need to be extensible and rich enough to allow fine grain granular policy statements to be expressed regarding the conditions that need to hold in order to distribute elements, the actions that are taken if rules fire and the transformations that need to be applied prior to returning information.
– Per-default no context information is distributed. This mechanism ensures that no information is provided to other entities without explicit indication. It also ensures that unable authorisation policies do not harm the context provider's privacy. This situation can occur when policies are stored centrally and need to be retrieved in order to compute the authorisation decision. Evaluating additional policies can only lead to more information being disclosed rather than less. As a consequence, exceptions cannot be expressed conveniently. We do not see this as a hard restriction since ephemeral identifiers are heavily used in Ambient Networks and can therefore be exploited to bypass blacklists by generating a new identifier. The unsuitability of the blacklist approach has been recognised with the current email system where new email addresses can be generated easily and they can also be spoofed.
– Simple conflict resolution strategy to prevent unintentional information disclosure. Configuring policies is an error prone task. In order to avoid misconfiguration and a false belief in the security of the system the policies need to be designed as simple as possible. This aspect is, to some extent, related to the user interface design but indirectly to the functionality provided by the underlying authorisation policy language.
– Focus on a distributed architecture. When context information is forwarded from one node to another within the network, where authorisation decision might take place at multiple locations and with authorisation policies being distributed at different places it is necessary to ensure that (a) policies travel with the context information either via reference or value and (b) the authorisation framework is able to deal with failure cases and dynamically changing policies.

4 Ensuring Privacy in the ContextWare Architecture

Given the high-level goals listed in the previous section a strong relationship with the work in the IETF GEOPRIV working group [3] is apparent. To avoid

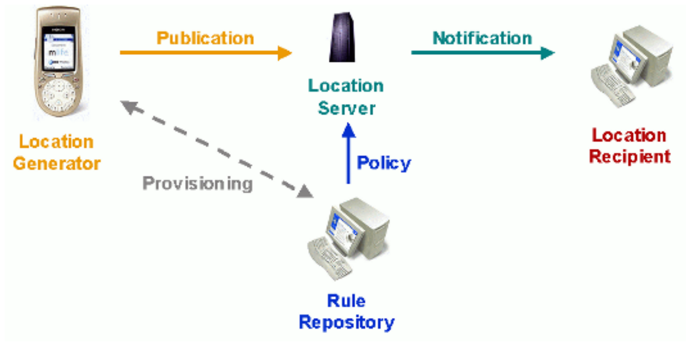


Fig. 2. GEOPRIV Architecture

reinventing a new privacy framework, we suggest to reuse the GEOPRIV privacy framework and to apply it to the ContextWare architecture in order to control access to the information stored in or referenced by the Context Information Base.

The GEOPRIV framework aims at protecting location information against unauthorised access to preserve the privacy of the owner of that information. Although GEOPRIV is mainly focused on authorisation of location information the framework was extended to provide authorisation functionality also for presence information as well. With RFC 3693 [9], a protocol-independent model for access to geographical location information was defined and is illustrated in Figure 2. The model includes a Location Generator (LG) that produces Location Information (LI) and makes it available to the Location Server (LS). A Rulemaker (RM) uploads authorisation policy rules to the Location Server. An authorisation policy is a set of rules that regulate an entity's activities with respect to privacy-sensitive information such as location information. Each rule consists of a condition, action and a transformation part. The Location Server (LS) stores LI and authorisation policies and acts as an entity that authorises access to a particular Target. The request sent by a Location Recipient (LR) to access location information of a particular Target and either receives an error message (in case of an authorisation failure) or the Location Object that consists of LI together with authorisation policies. Note that the Target is not shown in Figure 2 since its relationship to the other entities might vary depending on the given scenario.

In many scenarios some of the logical roles (such as LG, LS, Target, LR and RM) might be co-located into a single entity. For example, the Location Generator, the Location Server and the Target might be co-located at the same entity. The Target may also be the Rulemaker in many cases.

Figure 3 shows an attempt to map the existing GEOPRIV framework onto the ContextWare architecture created for storing context information in Ambient Networks. The subsequent paragraphs provide the details of this mapping. As illustrated above, a simplified description of the ContextWare architecture consists of the following main components [8]:

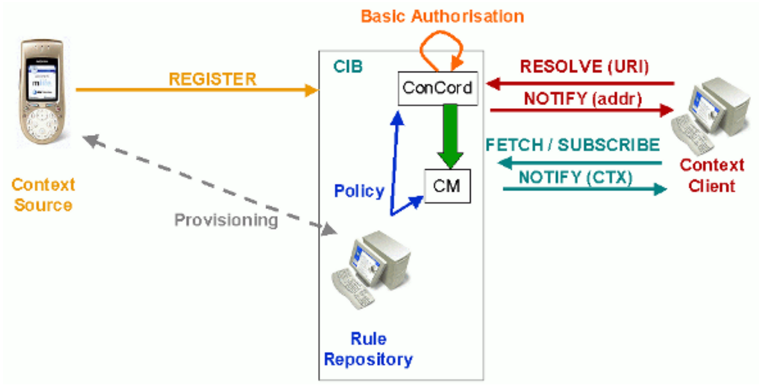


Fig. 3. Mapping GEOPRIV onto ContextWare architecture

- Context clients request context information
- Context sources provide context information
- Context coordinator/manager

When a client wants to access context information, it sends a request to the ConCoord. Note that an Ambient Network may have more than one ConCoord and might even push this functionality to the end hosts itself. The exchange of context information between ConCoords might utilise P2P filesharing techniques in very dynamic environment with a frequently changing network topology where nodes join and leave on a regular basis (such as networks that are created in an adhoc fashion). A context source registers with the ConCoord which raises the question of the used identifier for registration. To support a dynamic environment where identifiers are generated on the fly without prior registration it seems to be reasonable to utilise cryptographic identifiers. These identifiers can be generated by each node and allow the owner to show ownership of a particular identifier by using the corresponding private key, for example as part of a digital signature. A request by a client for context information can be relayed to the source providing the desired context information or can also be responded by the Context Manager.

Because of the nature of context information and the need for flexibility, ContextWare accounts for several scenarios requiring different aspects of access control to be considered:

1. Context information is stored "by reference" (e.g. it resides at the source and a reference is stored at the ConCoord). This requires a coarse access control policy to be enforced at the ConCoord (based on high-level agreements between domains of the client and of the queried ConCoord). More detailed policies are then stored and enforced directly at the source or on behalf of the source.
2. Context information is stored "by value" (e.g. it is copied at the CIB). This choice may be dictated by availability reasons (source of info not always

online) or by the willingness for the context source to delegate the enforcement of access control policies to the Context Manager. The context information may also be generated within the CIB, for example through aggregation processes implemented by the Context Manager. In both these cases coarse access control is enforced at the ConCoord as above, whereas the Context Manager takes on the responsibility of enforcing access control policies on behalf of the source it represents.

3. Context information can be *pulled* by a client several times using the same procedure that characterises a webpage download (i.e. lookup ConCoord once and use cached info for as long as the specified Time-To-Live (TTL) to address the source for subsequent queries). This avoids unnecessary involvement of ContextWare upon multiple retrievals but it assumes that the authorisation must be valid for at least the specified TTL. Other mechanisms might be needed to make the cached information invalid in the occurrence of events (i.e. de-merging Ambient Networks) that would affect access control policies.
4. Context information can be *pushed* through the means of *context associations* where the client and the source agree on the modalities of how such updating events take place (i.e. regular push, push if change occurred, protocols to use etc.).

The first two scenarios influence *where* access control policies are stored and executed. The third and fourth scenarios instead clarify *how* access control is implemented. It should be noted that not all the queries from context clients need to go through ConCoord or CM. *Context associations* are means to distribute context without necessarily involving either the ConCoord (apart from "signalling" phase) or the CM (apart from when the context source delegates the CM to act on its behalf).

Since authorisation policies are attached to the context information when retrieved, further distribution can be allowed, restricted or denied. Some degree of trust has to be placed on the entity that is granted access to context information in order not to violate the provided authorisation policies.

A client can request context information not only within its own domain but also outside it. Authorisation policies can restrict distribution within a domain or outside based on the structure of the identifier or based on various traits and roles. Analogously, a context source might register with a ConCoord of another domain; in this case the context source is external to the ConCoords domain. As an example, a roaming user makes its user profile available to the remote network.

Compared to GEOPRIV where the location server always acts as a Policy Enforcement Point, we have identified a similar role in the Context Manager, which is also responsible for setting up multipipes [8] for context aggregation. As illustrated above though, we are also adding the flexibility of granting the source of the context information itself to act as an enforcement point.

Also it is beneficial to consider having two levels of authorisation, one for coarse access control managed by the ConCoord and based on basic policies of the domain it represents (common to all context information), the other for refined access and based on finer grain policies implemented by the various sources. Note that either of those may be void if there are no concerns in sharing context information widely.

5 An Example of a Secured Context Request

Based on the above architecture and discussion, Figure 4 illustrates the sequence of events that characterises a *secured* context request. This scenario also covers the case whereby *context associations* are established between a context source and a client when multiple queries from the client are envisaged (such as desired in a subscription based environment).

First, the context source registers the URI for its context information with the ConCoord, it also notifies whether or not it wants to delegate the Context Manager as a proxy to deal with client requests (CM=YES in the case considered). Next, a context client asks (1. Fig.4) the ConCoord where the context information identified by the RESOLVE URI is located. The ConCoord performs (2a. Fig.4) the basic policy check. This basic authorisation (based on a common policy for all the context requests) provides the opportunity for the Ambient Network in which the ConCoord is to decide who to provide this "context lookup" service to. For example, a provider may want to deny access to context information for all foreign clients. The query is then resolved (2b. Fig.4), and the address of where the requested context is located (CM_addr in this case) is sent back to the client.

The client has now the address of the CM. It issues a FETCH request to pull the information or alternatively, it can establish a *context association* issuing a SUBSCRIBE request. The Context Manager, acting as a proxy in this example, checks whether the requested information is locally available. If not, it recursively

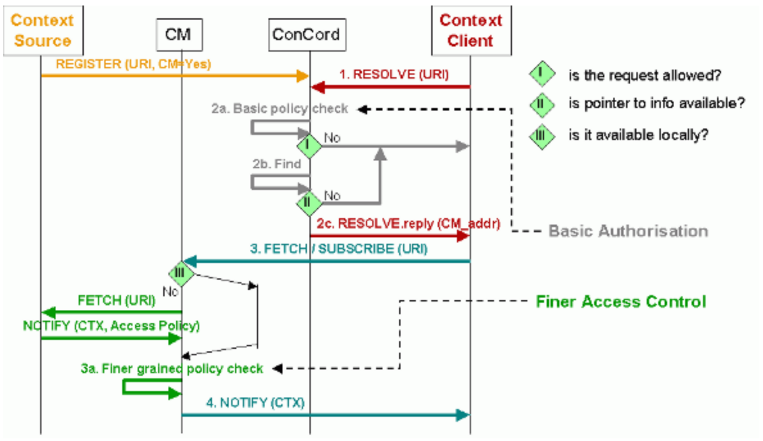

Context Source
CM
ConCord
Context Client
REGISTER (URI, CM=Yes)
1. RESOLVE (URI)
I is the request allowed?
II is pointer to info available?
III is it available locally?
2a. Basic policy check
No
2b. Find
No
2c. RESOLVE.reply (CM_addr)
Basic Authorisation
3. FETCH / SUBSCRIBE (URI)
No
FETCH (URI)
Finer Access Control
NOTIFY (CTX, Access Policy)
3a. Finer grained policy check
4. NOTIFY (CTX)

Fig. 4. Example Message Flow

fetches the information from the original source, authenticating itself as a context manager. The source responds with a NOTIFY message containing the context information and its access policies. The CM then checks the finer grained policy for the requested information on behalf of the source and returns (NOTIFY) the context information to the client.

Note that further FETCH requests from this client will involve neither the ConCoord nor the context source, but only the context client and the CM.

As it can be seen, the behaviour following a context request is similar to what happens in the GEOPRIV architecture with the Context Manager playing the role of the Location Server (checking finer grained access control policies that might be set by the context source itself).

6 Conclusions

The content of this paper illustrates how to ensure proper authorization for access to and distribution of context information. The ContextWare architecture is designed to store context information in an Ambient Network. The need for such a privacy-sensitive solution was generated by the often cited concerns that relate to context-aware systems, which are relevant regardless whether it is *user* context only or, as it the case with context-aware networks, also *network* context being manipulated.

While revisiting the functionality provided by the ContextWare architecture to highlight its privacy requirements, we positioned existing protocols for information distribution arguing in favour of a more powerful approach, supporting subscription based techniques, P2P distribution, semantically richer languages for context description and allowing authorisation policies to be attached and executed.

To support the ContextWare architecture with these goals and given that a strong relationship with the work in the IETF GEOPRIV working group was apparent, we based the authorisation and privacy framework for context-aware networks on the GEOPRIV privacy framework. We also illustrated with an example how the whole framework is supposed to ensure privacy control when a client requests a particular context information.

Some of the choices made to detail the sequence of events characterising a secured context request (e.g. two-levels authorisation, separation between the operations of the ConCoord and the CM etc.), were made purely based on common sense. This was based on considerations about how to minimise the control traffic and evenly distribute functionality load between the building blocks of the ContextWare architecture. Further work is needed to produce a more detailed performance analysis in support of such decisions. Also the details of how the policies are structured and executed were considered out of the scope of this paper and further work will need to be done in this direction to achieve greater understanding of performance of such a framework. This further detailed work could include a study on how to extend the GEOPRIV Common-Policy authorization framework [13] to perform authorization decisions for network specific state information (in the same fashion as exercised with [14]).

We also highlighted the requirement for an aggregation capability to be performed by the Context Manager to "create" context information through interpretation of low level information available i.e. through sensors. This transformation capability opens a whole new space where elicitation of privacy policies might become very complex considering that context processors or aggregators might have to deal with conflicting policies from the sources of low level context information. This aspect will also need to be addressed with further work.

Acknowledgements

This document is a byproduct of the Ambient Networks Project, partially funded by the European Commission under its Sixth Framework Programme. It is provided "as is" and without any express or implied warranties, including, without limitation, the implied warranties of fitness for a particular purpose. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Ambient Networks Project or the European Commission.

References

1. 3GPP TS 23.141 "Presence service; Architecture and functional description; Stage 2", Version 6.7.0, 2004-09-27, available at http://www.3gpp.org/ftp/Specs/html-info/23141.htm (April 2005).
2. "Ambient Networks Project", available at http://www.ambient-networks.org, (April 2005).
3. Geographic Location/Privacy (geopriv) Working Group Charter, available at http://www.ietf.org/html.charters/geopriv-charter.html, (April 2005).
4. J.I. Hong and J.A. Landay. An Architecture for Privacy-Sensitive Ubiquitous Computing. In Proceedings of The Second International Conference. In Proceedings of The Second International Conference on Mobile Systems, Applications, and Services (Mobisys 2004). Boston, MA. To Appear 2004.
5. A. Karmouch, A. Galis, R. Giaffreda, T. Kanter, A. Jonsson, A. M. Karlsson, R. H. Glitho, M. Smirnov, M. Kleis, C. Reichert, A. Tan, M. Khedr, N. Samaan, L. Heimo, M. El-Barachi, J. Dang: Contextware Research Challenges in Ambient Networks. MATA 2004: 62-77.
6. M. Khedr et al., "Negotiating Context Information in Context-Aware Systems," IEEE Intell. Syst., vol. 19, no. 6, pp. 21-29, Nov./Dec. 2004.
7. N. Niebert, et. al.: Ambient Networks – An Architecture for Communication Networks Beyond 3G, IEEE Wireless Communications (Special Issue on 4G Mobile Communications - Towards Open Wireless Architecture) (2004).
8. C. Reichert, M. Kleis, R. Giaffreda, "Towards Distributed Context Management in Ambient Networks", Forthcoming publication to be presented in the Session on Service Access and Mobility of the 14th IST Mobile and Wireless Communications Summit, Dresden, Germany, 19-23 June 2005.
9. J. Cuellar, J. Morris, D. Mulligan, J. Peterson and J. Polk: "Geopriv Requirements", RFC 3693, February 2004.

10. B. N. Schilit, N. I. Adams, and R. Want. "Context-Aware Computing Applications." In Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, December 1994. Pages 85-90. IEEE Computer Society.
11. A. F. Westin, Privacy and Freedom, (New York: Atheneum: 1967) p. 7.
12. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler: "SIP: Session Initiation Protocol", RFC 3261, June 2002.
13. H. Schulzrinne, J. Morris, H. Tschofenig, J. Cuellar, J. Polk and J. Rosenberg: "A Document Format for Expressing Privacy Preferences", Internet-Draft (work in progress), draft-ietf-geopriv-common-policy-03.txt, February 2005.
14. H. Schulzrinne, H. Tschofenig, J. Morris, J. Cuellar and J. Polk: "A Document Format for Expressing Privacy Preferences for Location Information", Internet-Draft (work in progress), draft-ietf-geopriv-policy-05.txt, November 2004.
15. Bellwood, T., Claement, L. and C. von Riegen, "UDDI Version 3.0.1, UDDI Spec Technical Committee Specification, Dated 20031014", October 2003.

Widget Integration Framework for Context-Aware Middleware

Eun-Seok Ryu, Jeong-Seop Hwang, and Chuck Yoo

236, Department of Computer Science and Engineering, Korea University,
Anam-dong, Sungbuk-ku, Seoul, S.Korea
{esryu, jshwang, hxy }@os.korea.ac.kr

Abstract. Widget Integration Framework (WIF) is a framework that covers diverse widgets and their interactions with higher layers above. The framework offers several advantages of supporting a programming abstraction over widgets, supporting high reliability, run-time widget binding to the middleware and augmenting a service discoverer with available widget state information. This paper explains the processes of designing the WIF, including implementation, and applying WIF to the middleware. As an example, we explain a location-based service that uses a location positioning widget in a middleware employing WIF.

1 Introduction

There are several researches [15,16,17,18] for sensing environment changes and offering the most suitable context information to users. However, they were designed and implemented depending on restricted situations, only in several scenarios, only focusing on sensing an environment and managing context information with service discoverers and context manage modules, and are heavy to be applied to the mobile device for the future light-weight mobile middleware.

This paper presents WIF that addresses how middleware integrates and manages widgets effectively, binds appropriate widgets to the service discoverer, and supports mobile environment. The goal of WIF is to design and implement a light-weight widget abstraction layer for pervasive middleware. The focus is on widget integration, and such integration requires a programming abstraction over widgets. This framework encapsulates widgets, transmits sensed context information from widget to the upper middleware layers, registers and deregisters widget in real time and checks the registered widget's state for service discoverer.

The WIF has following advantages. First, it is light. Existing researches for pervasive environment(e.g. UIUC's Gaia and ASU's RCSM) provide CORBA-based services. However, this way makes offering dynamic service to handheld devices hardly. WIF is designed to support light-weight context-aware middleware platform for a mobile device. Second, WIF helps service discoverer to select a proper widget in the middleware by using our SDP(Service Discovery Protocol)-like protocol. Because WIF has an internal mechanism that can find available widgets and that delivers these widgets to the upper middleware layer, dynamic reconfiguration is supported. Third, WIF makes a middleware application fault-tolerant. Current middleware applications may impose a fatal obstacle on the entire system caused by an error from various hardware sensors. WIF monitors error occurrences by administrating widgets and disallowing the widget to operate in services. Through this, the WIF can support strong fault-tolerance. Finally, WIF has excellent interoperability with other middleware layers. It can interact effectively with other modules in middleware using the *Service Interaction Broker*[20] offering basic communications as well as an interface for interaction between service objects. The WIF that was developed through our research and related individual functions was verified by actual demonstration.

This paper is composed as follows. Section 2 explores existing systems aimed to support pervasive environments with their own visions. In Section 3, we describe the goal of this paper and the WIF design considerations. Section 4 describes the design of the WIF and its components in detail. In Section 5, we show how this WIF works in real pervasive environments by applying it to the middleware. In Section 6, we finally conclude our framework research for active surroundings and propose practical usage.

2 Related Work

2.1 Middleware for Pervasive Computing

Thorough research regarding middleware application that support pervasive environments includes projects, GAIA from the University of Illinois at Urbana-Champaign, AURA from Carnegie Mellon University, Oxygen from MIT and Dey's Context Toolkit from Georgia Institute of Technology. This paper addresses each of these research projects.

2.1.1 GAIA - UIUC

GAIA[1,2,3,15] is intended to coordinate software entities and heterogeneous networked devices contained within a physical space. GAIA is composed of four major building blocks, the Gaia Kernel, the Gaia Application Framework, the QoS Service Framework, and the Applications. The Gaia kernel contains a management and deployment system for distributed objects and an interrelated set of basic services that are used by all applications. The Application framework provides mobility, adulteration, and dynamic binding. The QoS service framework not only provides resource management for the QoS sensitive applications, but also dynamically adapts applications based on the QoS constraints and determines appropriate nodes for service instantiation. However, small mobile devices cannot cooperate autonomously without the infrastructure.

2.1.2 Aura - CMU

Aura[4,5,16] aims to provide a distraction-free computing environment where people can get services or perform their jobs without interventions from the system or environments. Aura is composed of five main components, Intelligent networking, Coda, Odyssey, Spectra and Prism. Odyssey supports resource monitoring and application-aware adaptation, and Coda provides support for nomadic, disconnect-able, and bandwidth-adaptive file access. Spectra is an adaptive remote execution mechanism that uses context to decide how to best execute the remote call. And, Prism is responsible for capturing and managing user intent.

2.1.3 Oxygen - MIT

Oxygen [17] aims to provide human-centered computing environments which help people automate repetitive human tasks, control a wealth of physical devices in the environment, find the information people need, and enable us to work together with others through space and time. To support dynamic and varied human activities, the Oxygen system focuses on the pervasiveness of the system or devices, supporting nomadic users, and representing system adaptability.

2.1.4 Context Toolkit – Gatech

The Context Toolkit[10,18] provides designers with the abstractions they have described — widgets, interpreters, aggregators, services and discoverers - as well as a distributed infrastructure. The Context Toolkit was developed using the Java programming language, programming language independent mechanisms were used, allowing the creation and interoperability of widgets, interpreters, aggregators and applications in any language.

2.2 Protocols for Service Discovery

Service Discover Protocol(SDP) finds a way software and network resources are configured, deployed, and advertised, all in favor of the mobile user researches on the emerging technologies of service discovery in the context of wireless and mobile computing are increasingly important. Some of these emerging SDPs include SLP, Jini, and UPnP(Universal Plug and Play). In these SDPs, UPnP uses SSDP(Simple Service Discovery Protocol)[23] for service discovery for announcing a device's presence and capabilities to others as well as discovering other devices or services.

3 Key Considerations

WIF is an individual framework that has ability to support applications without any other components or resources.

WIF is located in middleware's bottom layer in whole hierarchical system layers by integrating widgets which wrap sensors and it interacts with upper middleware through SOAP-based interface S*ervice Interaction Broker*[13]. If context-aware middleware uses WIF in it, it may manage service discoverer just only without caring about managing all sensors. In this chapter, we describe the key considerations of designing WIF.

- **High reliability** : The application expects data received from middleware layer to be reliable. Therefore, to support reliability, the middleware application needs functions that manage a widget's registration and deregistration, finds error occurrences and ignores targeted modules. This also includes filtering of abnormal sensor values.

- **Dynamic binding in mobile environment** : To support several services required from applications, middleware should be reconstructed dynamically according to situations of the context and of available hardware. Therefore, WIF needs to support run-time widget binding so that middleware can act plug-and-play. Some of previous systems which were developed for general purposes are heavy to apply to mobile device, because they are using XML-based protocol and CORBA for general interaction. However, we are trying to design a light-weight framework by choosing efficiency instead of generality. Of course, this way may cause system dependency to WIF, because it has to know all widgets will be used and has functions of those widgets in advance.

- **Common interface over different widgets** : WIF has interoperability with other middleware layers. It can interact effectively with other modules in middleware using the *Service Interaction Broker(SIB)*[20] offering basic communications as well as an interface for interaction between service objects. By this reason, other middleware components are allowed to just using a SIB-level interface. Consequently, it makes WIF offer a common interface over different widgets.

- **Augmenting a service discoverer by offering status information of available widgets** : If applications request necessary services from a middleware application's Service Discoverer, the Service Discoverer must be able to find and deliver available widgets among whole widgets, which support required services.

4 Proposed System Architecture

Fig. 1 shows the WIF system architecture and its internal components. It consists of Widget Information Manager, Fault Detector, Service Matcher, Widget Register and Widgets. In this section, each component is explained.

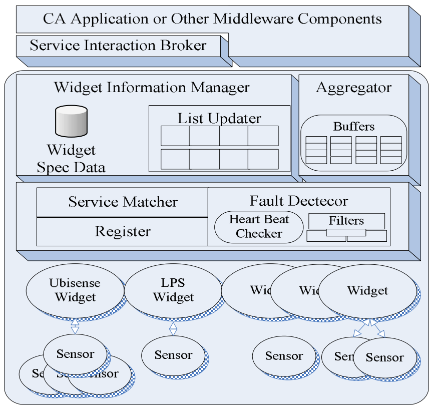


Fig. 1. Widget Integration Framework (WIF) Architecture

4.1 Widget

We use the concept *widget* as the Context Toolkit. Context widgets encapsulate information about a single piece of context, such as location or activity, for example. They provide a uniform interface to components or applications that use the context, hiding the details of the underlying context-sensing mechanisms [24]. In our research, we implemented Ubisense widget for location based service and it is treated in section 5.2.

4.2 Widget Information Manager

Widget Information Manager (WIM) role is to manage widget state information. It manages state information of widgets received from the Widget *Register* or *Fault-Detector* with *List updater*. It also keeps specification information of several widgets in order to find a suitable widget through *Service Matcher* regarding services required from the *Service Discoverer*. Also it exchanges all information from widgets with other modules in the middleware application through *Service Interaction Broker*.

4.2.1 List Updater

List updater (LU) belongs to WIM and manages the widget status information table. *Widget Type Number* in Table 1 divides widgets by the core unit role. For example, number 0 means that those widgets have relations with the Location Positioning System and number 1 means that those widgets are related to the sound/video input system. Within such classification, the *widget number* is granted to the widget, which is actually installed or is going to be installed. Then, the *Widget Type Number* and the *Widget Number* are combined and used as *Widget ID*. LU communicates whether relevant widget operates, encoded as 1 or 0 to the *Live Information* field. And, *State Information* expresses the state of widget (1: Occupied, 2: Requested, 0: available (non-used)). That is, value 0 means that the widget is not used and required at all, value 1 means that it is being used and value 2 means it was required to use if the relevant one is linked to the system. LU's role is table management. Therefore, other modules can use the necessary widget or unavailable widget in services using this table representation.

Table 1. Example of Widget State Table

Widget Type Number	Widget Number	Live Information	State Information
0	001	1	1 (Occupied)
(Location widgets)	002	1	1 (Occupied)
1	001	1	1 (Occupied)
(Sound Level /	002	0	2 (Requested)
Camera widgets)	003	0	0 (Available)
2	…	…	…
(…)	…	…	…

4.3 Fault Detector

Fault Detector is designed to check the state of the widget through a periodic heartbeat checker. It updates the *Live Information* value and the *State Information* value regarding the widget state table in *Widget Information Manager* if an error happens to the sensor or widget. LU informs *Service Discovery Manager* of these events. This guarantees high reliability by removing problematic widgets or replacing them with a new one. *Fault Detector* has various filters. Data, which is input from sensors, is transferred from the widget, and filtered. For example, a position revision filter corrects context location information retrieved from the Ubisense widget.

4.4 Service Matcher

Service Matcher finds a suitable widget for services that are required from *Service Discoverer*. For example, *Service Discoverer* in middleware applications requests a user's position information service from applications and must understand those requests, find a suitable widget and bind to it. *Service Matcher* references the widget state table as in Table 1 and finds an available widget, which can service position information. Detailed process steps are explained in Section 4.6.

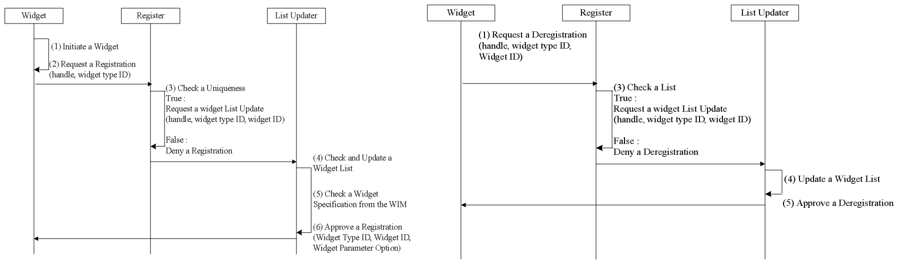
4.5 Widget Register

Widget Register takes charge of registration and deregistration of a widget. The widget helps *Widget Information Manager* by passing the state event to LU when it is registered/deregistered to the system. Though other research systems receive data only from pre-registered widget, Widget Register can support dynamic environment where widgets are registered/deregistered in real time by accepting a new incoming widget and notifying it to *Widget Information Manager*. In the result, *Widget Register* makes WIF bind a new incoming widget.

4.6 Interaction Diagram Among Components

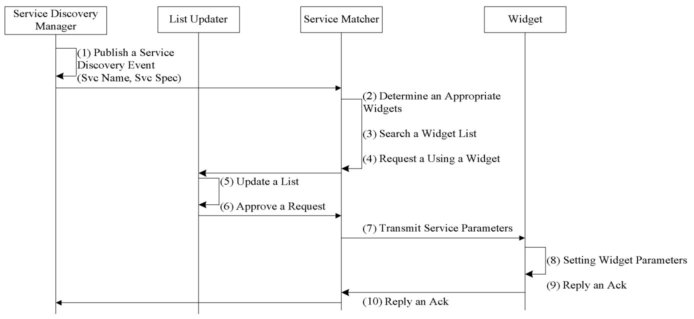
The components of the WIF run through frequent interaction among them. This section explains 3 cases to show dynamic binding process using interaction diagrams [11] below.

(1) Widget registration and deregistration



Widget
Register
List Updater
(1) Initiate a Widget
(2) Request a Registration (handle, widget type ID)
(3) Check a Uniqueness
True : Request a widget List Update (handle, widget type ID, widget ID)
False : Deny a Registration
(4) Check and Update a Widget List
(5) Check a Widget Specification from the WIM
(6) Approve a Registration (Widget Type ID, Widget ID, Widget Parameter Option)
Widget
Register
List Updater
(1) Request a Deregistration (handle, widget type ID, Widget ID)
(3) Check a List
True : Request a widget List Update (handle, widget type ID, widget ID)
False : Deny a Deregistration
(4) Update a Widget List
(5) Approve a Deregistration

(2) Widget assignment by service request (when needed widget was pre-registered)



Service Discovery Manager
List Updater
Service Matcher
Widget
(1) Publish a Service Discovery Event (Svc Name, Svc Spec)
(2) Determine an Appropriate Widgets
(3) Search a Widget List
(4) Request a Using a Widget
(5) Update a List
(6) Approve a Request
(7) Transmit Service Parameters
(8) Setting Widget Parameters
(9) Reply an Ack
(10) Reply an Ack

Fig. 2. Interaction Diagram among Components

(3) Widget assignment by service request

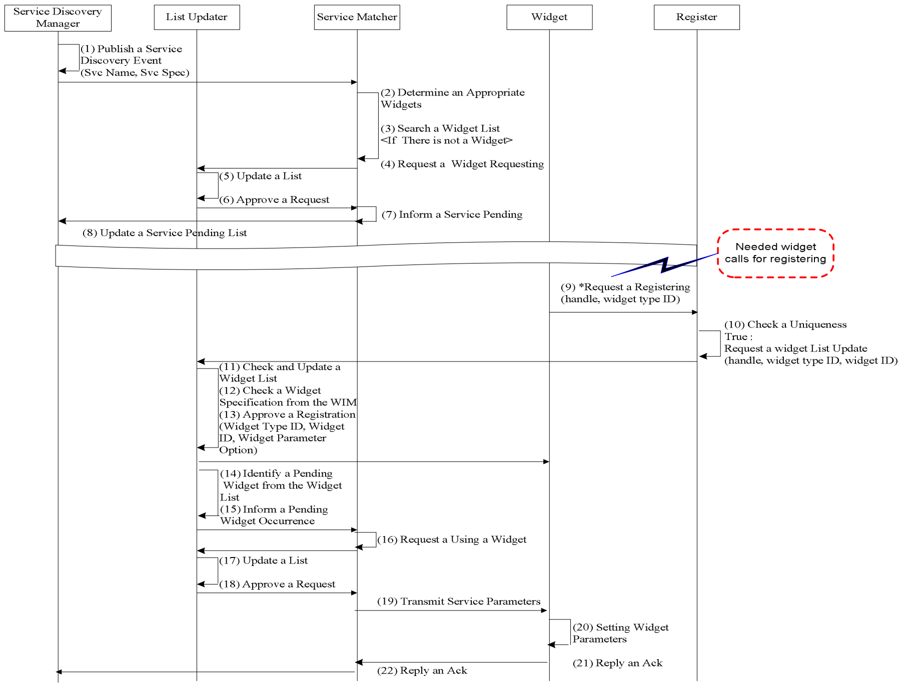

Service Discovery Manager
List Updater
Service Matcher
Widget
Register
(1) Publish a Service Discovery Event (Svc Name, Svc Spec)
(2) Determine an Appropriate Widgets
(3) Search a Widget List <If There is not a Widget>
(4) Request a Widget Requesting
(5) Update a List
(6) Approve a Request
(7) Inform a Service Pending
(8) Update a Service Pending List
Needed widget calls for registering
(9) *Request a Registering (handle, widget type ID)
(10) Check a Uniqueness True : Request a widget List Update (handle, widget type ID, widget ID)
(11) Check and Update a Widget List
(12) Check a Widget Specification from the WIM
(13) Approve a Registration (Widget Type ID, Widget ID, Widget Parameter Option)
(14) Identify a Pending Widget from the Widget List
(15) Inform a Pending Widget Occurrence
(16) Request a Using a Widget
(17) Update a List
(18) Approve a Request
(19) Transmit Service Parameters
(20) Setting Widget Parameters
(21) Reply an Ack
(22) Reply an Ack

Fig. 3. *(continued)*

5 Applying the WIF to Middleware

In this section, we address how to apply WIF to the middleware for a location-based service to show its usability. We currently developed several widgets including the *Ubisense widget*, *Fault-Detector* with filters for correcting location information, and implementing remainder parts according to our design. And our WIF is applied to the group-aware middleware[20] that was developed by ICU[13] to manage low level widgets. In the following sections, we briefly describe the group-aware middleware.

5.1 Group-Aware Middleware

Group-Aware Middleware(GAM) is researched to make middleware understand group-context in ubiquitous home. And this middleware infrastructure for *Active Surrounding*(a kind of ubiquitous home)[14] that focuses on group-context awareness and processing is studied by the pervasive computing group of Information and Communications University[11, 12, 13]. They defined *Active Surrounding* as a pervasive environment where entities (devices or services) actively response to user actions or help users to perform their jobs without intrusion to the users. We tried to apply our WIF to this GAM implementation so that it manages all sensor-related works. As a result, GAM with WIF consists of 4-units (*Context Manager, Dynamic*

Reconfiguration Supporter, Context-aware Service Discoverer and *Widget Integration Framework*(WIF)). Each component communicates with other components in the form of platform and location independence by the *Service Interaction Broker.*

Whole group-aware ubiquitous middleware research consists of unification of research results. The CAService(Context-Aware Service) can merge with required context information and publish events through the *Service Interaction Broker* API or join other events using the *Context Management* API by *Active Surroundings* middleware. *Context Management* relates to context information registered from the CAService by various sensor inputs that are input from the WIF. Also, it delivers the relevant CAService using event appearance context information. The *Dynamic Reconfiguration Framework* delivers events drawn from the CAService to CAServices that join to the events. In this situation, if it cannot find the appropriate CAService it conducts a semantics search of services through context-aware *Service Discovery*, searching for the CAService. Context-aware *Service Discovery* evaluates requests from the *Dynamic Reconfiguration Framework* and the CAService based on semantic characteristics of registered services. The Service Discover returns, finding the most suitable service to present context. The WIF focused in this paper integrates and manages widgets and allocates suitable widgets according to the service request.

5.2 Location-Based Service by Ubisense Widget

This paper explains location-based service by WIF-integrated middleware to show WIF's usability. This research uses Ubisense hardware sensor[21] devices to get context location information and develops Ubisense widgets for controlling those sensors. The characteristics of our implemented Ubisense widget are as follows.

- Passes location information (ID, timestamp, x, y, z) to the *Service Manager*
- Supports interaction between *Service Manager* and widgets
- Supports platform independence using JNI (Java Native Interface)

In the following sections, we examine the Ubisense widget embodied in this research in detail. For an explanation for the Ubisense marketing product, refer to the web site, this paper will not handle this in detail. Ubisense could be used regardless of development language because it supports COM (Component Object Model). Our middleware was developed using the Java programming language, because of this, we put a Java-COM wrapper over the Ubisense COM and developed the widget over the top. This Ubisense widget delivers sensed data to the upper middleware layers using a requested ratio from the middleware application. The system can achieve registration/deregistration of specific widget requests. Also, it corresponds to the Fault-Detector's heartbeat checking which is explained from lower middleware layers and has functions that transmit live messages. Of course, this ubisense widget achieves status management and message delivery with interaction using several components in the WIF. Context information is exchanged with other frameworks of a group-aware middleware using *Service Interaction Broker.*

The next figure shows the concept that the Ubisense widget delivers sensing information to upper middleware layer service manager using the *Service Interaction Broker (SIB)* and SOAP (Simple Object Access Protocol) interface.

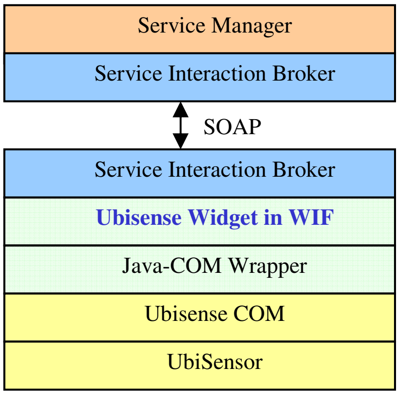


Fig. 4. The Hierarchical Architecture for Ubisense Widget

The Ubisense widget role is to deliver this context information to the upper layer by chasing tag-attached person's position from the actual Ubisensor. Therefore, it supports user's position chase, which is a basic requirement for all ubiquitous environments. This system was installed and demonstrated with sensors in a practical *'Active Home*' implementation. The demonstration scenario is that *'a group-aware middleware'* turns into a TV when user stands in front of it, shows broadcasting what a user wants and changes indoor temperature and lighting for more than two persons. The video clip for this demonstration is referenced in this paper [19].

6 Conclusion and Discussion

This paper explained several advantages that context-aware middleware can achieve by managing widgets that wrap sensors effectively. We describe the process of designing and implementing framework internals for managing widgets that support these advantages.

Many current middleware applications support a situation-aware application, integrating and analyzing primitive sensed information from widgets and delivering these to the middleware upper layer. However, these middleware applications cannot correspond properly in situations of widget malfunctioning and registering/deregistering during their operation. Therefore, we designed and implemented the *Widget Integration Framework* (WIF), located in the middleware lower layers. This framework encapsulates widgets, transmits sensed context information from widget to the upper middleware layers, registers/deregisters widgets in real time and checks a registered widget's state. With this processing, the WIF has exceptional error resilience properties and widget dynamic binding properties. The framework also has good interoperability because it communicates context information with other components of entire middleware applications *through the Service Interaction Broker (SIB),* using a SOAP interface. In addition, this framework is independent of middleware platforms and can be deployed using any development programming language on any operating system.

References

1. Manuel Román, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces", IEEE Pervasive Computing, pp. 74-83, Oct -Dec 2002.
2. Christopher K. Hess, Manuel Roman, and Roy H. Campbell, "Building Applications for Ubiquitous Computing Environments", In International Conference on Pervasive Computing (Pervasive 2002), pp. 16-29, Zurich, Switzerland, August 26-28, 2002.
3. Renato Cerqueira, Christopher K. Hess, Manuel Roman, Roy H. Campbell, "Gaia: A Development Infrastructure for Active Spaces", In Workshop on Application Models and Programming Tools for Ubiquitous Computing (held in conjunction with the UBICOMP 2001), September 2001.
4. David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste, "Project Aura: Towards Distraction-Free Pervasive Computing", IEEE Pervasive Computing, special issue on Integrated Pervasive Computing Environments, Volume 1, Number 2, April-June 2002, pages 22-31.
5. Sousa, J.P., Garlan, D., "Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments", Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture 2002, Montreal, August 25-31.
6. S. S. Yau, F. Karim, Y. Wang, B. Wang, and S. K. S. Gupta, "Reconfigurable Context - Sensitive Middleware for Pervasive Computing" IEEE Pervasive Computing, July-September 2002, IEEE Computer Society Press, Los Alamitos, USA, pp. 33-40.
7. Sergio Marti and Venky Krishnan, "Carmen: A Dynamic Service Discovery Architecture", Technical Report, August 2002.
8. Andry Rakotonirainy, Jaga Indulska, Seng Wai Loke, and Arkady Zaslavsky, "Middleware for Reactive Components: An Integrated Use of Context, Roles, and Event Based Coordination", Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms, pp. 77-98, Heidelberg, Germany, November 12-16, 2001.
9. W. Zhao, H. Schulzrinne, E. Guttman, C. Bisdikian, W. Jerome, IETF RFC 3421, "Select and Sort Extensions for the Service Location Protocol (SLP)", November 2002.
10. Anind K. Dey, Daniel Salber and Gregory D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Human-Computer Interaction (HCI) Journal, Volume 16 (2-4), 2001, pp. 97-166.
11. Insuk Park, Soon Joo Hyun, Donman Lee, "Context-Conflict Management for Context-aware Applications", Ubiquitous Computing Workshop 2004.
12. Dongman Lee, Soon J. Hyun, Young-Hee Lee, Geehyuk Lee, Seunghyun Han, Sae-Hoon Kang, Insuk Park, and Jinhyuk Choi, "Active Surroundings: A Group-Aware Middleware for Ubiquitous Computing Environments", Ubiquitous Computing Workshop 2004.
13. Pervasive Computing Group, "A Middleware Infrastructure for Active Surroundings", TR-CSPG-2003-004-28-002. http://cds.icu.ac.kr/druid/res/TR-CSPG-2003-004-28-001.pdf
14. Dongman Lee, "Active Surroundings: A Group-Aware Middleware for Embedded Application Systems", Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC '04).
15. Gaia, web site: http://choices.cs.uiuc.edu/gaia/
16. Aura, web site: http://www.cs.cmu.edu/~aura/
17. Oxygen, web site: http://oxygen.lcs.mit.edu/
18. Context Toolkit, web site: http://www.cs.berkeley.edu/~dey/context.html
19. WIF, web site: http://os.korea.ac.kr/mediateam/WIF.htm
20. The Final Report of the Operation Digital Media Lab, February 2005.

21. Ubisense Company : http://www.ubisense.net
22. Simple Object Access Protocol (SOAP) : http://www.w3.org/TR/soap/
23. Simple Service Discovery Protocol (SSDP) Internet Draft
: http://www.upnp.org/download/ draft_cai_ssdp_v1_03.txt
24. Yen-Wen Lin and Hsin-Jung Chang, "Service Discovery in Location Based Services for Low-Powered Mobile Clients", http:// jitas.im.cpu.edu.tw/2004-2/5.pdf
25. Anind K. Dey, Daniel Salber and Gregory D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Human-Computer Interaction (HCI) Journal, Volume 16 (2-4), 2001, pp. 97-166

Service Deployment in Active Networks Based on a P2P System

André Ribeiro Cardoso[1], Ahmed Serhrouchni[1], Bertrand Mathieu[2], and Mikaël Salaün[2]

[1] Télécom-Paris, LTCI-UMR 5141 CNRS,Paris, France
{acardoso, ahmed}@enst.fr
[2] FranceTelecom R&D, CORE/M2I, 2 Av. Pierre Marzin, 22300 Lannion, France
{bertrand2.mathieu, mikael.salaun}@francetelecom.com

Abstract. Active networks have been a research topic for years and many contributions have helped to refine the active node architecture, the active networks protocols and the needed functionalities. One of them is the service deployment, which is a key functionality for the success of active networks. Many research papers studied this item but how the active node information is known and used for the node selection is not satisfactory. In this paper, a new concept based on a Peer-to-Peer system, enhancing the deployment process, is presented. This P2P system is used by active nodes to advertise theirs capabilities, which enables "control active nodes" to perform the service deployment, depending on the service requirements and constraints. The concept of administrative domains is also taken into account, when deploying services, for widespread active networks. This work is related to the control plane of active networks, based on a P2P system and can be classified as a new approach for the service deployment.

1 Introduction

Active networks are networks composed of active nodes, located inside the network, that have the ability to perform processing tasks on flows passing through them. This enables to deploy new added-value network services or to enhance the quality of end-to-end application services. Services like active web cache, web service balancing, monitoring, congestion control, firewall, transcoding and so on are well suited to the active network environment. Processing tasks can be executed, when required, in one or several intermediate network node(s), meaning that this processing code may be dynamically deployed or removed from active nodes. However, most active networks deployment process requires an explicit specification of which nodes will be used by the application. This means a complete understanding of the underlying network infrastructure by the entity in charge of deploying the code (the administrators of the active networks, the service providers or the end-user), and so it inhibits the widespread use of active networks.

Peer-to-peer (P2P) systems may be defined as distributed systems consisting of nodes that can join and leave the (P2P) network at any time and that have equal capabilities and roles, without any centralized control. They provide interesting features such as wide-area routing architecture, efficient search of data items, redundant storage, permanence, fault-tolerance, self-organizing and massive scalability. Hybrid P2P systems, with some "super-peers" having extended role, have been designed for scalability and control issues.

Combining these two technologies, this paper presents a new concept, which based on a P2P system, enhances the service deployment in active networks. The concept of administrative domains has been taken into account in our solution, electing one active node as a "control active node", which is the contact point for other administrative domains. The P2P system is used by nodes and by "control active nodes" to advertise theirs capabilities inside one administrative domain and between different administrative domains. This work is related to the control plane of active networks, based on P2P systems and can be classified as a new approach for the service deployment.

This paper is structured as follows: Section 2 presents service deployment in active networks. Section 3 identifies the classes of P2P systems. The layered P2P and active network is described in section 4. Section 5 focuses on implementation issues. Finally, the future work and conclusions of this paper are presented in section 6 and 7.

2 Service Deployment in Active Networks

The difficulties in deploying new services in traditional IP networks led to the development of active networks [11]. The key idea is to dynamically deploy services, in the network, which perform processing tasks inside intermediate nodes. In order not to waste node resources, those services must be dynamically deployed when needed and removed when no longer necessary. Services, implemented in active networks, can be described as active packets or distributed programs (plug-ins extensions) that run in active nodes [1]. Although there have been several studies [2][4], deploying new services is not as simple as expected. As several active nodes become available, an efficient way to discover an active node capable of providing the desired services is required. According to [2], there are two levels for service deployment in an active network and a comprehensive framework must support them at both levels:

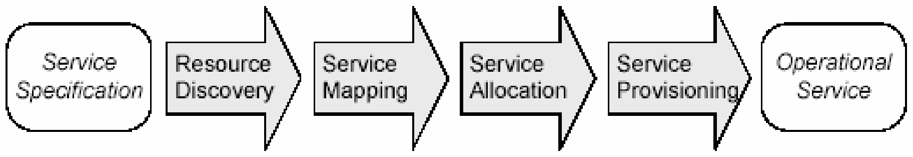


Fig. 1. Tasks for providing network services [4]

- The network level comprehends the identification of the nodes that match the service requirements and are suitable to execute the requested service.
- The node level consists of the installation and configuration of the appropriate service code within the node environment.

[4] states that providing a new service within an active network environment involves several tasks (Fig 1).

The *service specification* defines the components required by a service and describes how components are interrelated. *Resource discovery* identifies the location and capabilities of processing resources embedded within the network. *Service mapping* translates the service specification into the physical network graph while taking into account all service-specific constraints. The *service allocation* task reserves and configures appropriate physical resources. *Service provisioning* is the final task that executes all required components to provide an operational service for users [4].

In this paper, the focus is done on the resource discovery task and the mean to provide node information for the active node selection when deploying services.

3 Peer-to-Peer (P2P) Systems

Nowadays, three main classes of P2P systems may be identified: centralized, decentralized, and hybrid. The decentralized may still be divided into decentralized but structured and, decentralized and unstructured [6]. The main difference between them is the mechanism employed to look up resources in the P2P network.

In centralized P2P systems the description and location of resources are stored in a directory of a central server. Thus, nodes query this central server to find which node hosts the desired resource. The content always stays on the client side, never passing through the server. Such centralized approach does not scale well and has single points of failure.

Decentralized and unstructured P2P system has neither a central directory server nor rules to define or constrain the connection among nodes. By unstructured we mean that there is no control over the placement of files that stay in local nodes. To find a resource, a node queries its neighbors employing a flooding-based mechanism. Such an approach is not suitable for extensive networks since each node evaluates the query locally and if the desired resource is not present, queries must be sent to a lot of nodes. The lack of structure leads to scalability problems because each node must maintain a route to each other in the network. To limit the flooding, a mechanism like a TTL (Time To Live) is used. The drawback of this approach is that queries may not find a positive reply if the TTL expires before reaching the node hosting the resource. Gnutella [9] and Freenet [10] are some examples of this system class.

In decentralized but structured P2P system, the connection among nodes is tightly controlled and the resources are placed at specific locations. It employs the DHT (Distributed Hash Table) abstraction in the context of storing and retrieving data in the P2P network so that data object (or value) location information is placed deterministically, at the nodes with identifiers corresponding to the unique key data object. This structure enables efficient data discovery by using the given keys and is especially suitable for the development of large-scale network. Queries are forwarded across overlay links to nodes whose nodeIDs are progressively closer to the key in the identifier space. Content Addressable Network (CAN) [14], Chord [13], Tapestry [15], and Pastry [12] are the main examples of this system class.

Hybrid P2P systems combine the characteristics of both centralized and decentralized models. Decentralization means extensibility, fault-tolerance, lawsuit proofing, scalability, and so on. Partial centralization makes the system more consistent as there are relatively fewer hosts that holds authoritative data. Every user elects its super-peer, which is the "central server" for "local nodes" and can communicate with other super-peers. FastTrack/KaZaA , eDonkey are some examples of this system.

4 A Layered P2P and Active Network

In current works, service deployment is often closely coupled with the underlying active network technology. The proposed architecture is more open since the service deployment infrastructure can be deployed over any active network platform. This can be achieved through the use of a P2P layer over the active network layer.

Fig 2 depicts the layered P2P and active network. The active network layer, placed over the traditional network layer, is composed of two planes: the active network plane containing a subset of network layer nodes and the control plane assembling some of these nodes as control nodes of active domains. Above is the P2P layer, used for the service deployment, running in all active nodes.

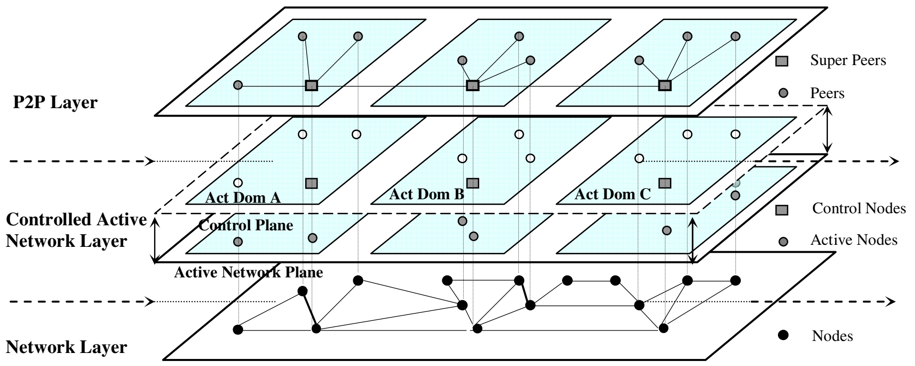


Fig. 2. The Layered P2P and Active Network

4.1 The Controlled Active Network Layer

As in [7] [16], this paper proposes to manage the active network and control the code loading by adding a control plane over the active network plane (Fig. 2).

- *Active Network Plane:* this plane gathers the standard active nodes. These nodes offer the basic primitives allowing access to the local resources and the processing of data passing through the node. They access the capsule fields (addresses and data), consult the routing tables or apply the routing functions to route the capsules according to their destinations, manage the resources assigned to the capsules, control their right to execution, and communicate with the control nodes.
- *Control Plane:* this plane gathers the control nodes in the active network. The control nodes are responsible for controlling the network, loading code into active nodes and registering new protocols. Code loading occurs either at the beginning of the application when the control node deploys the service code in all suitable

active nodes, or at the reception of an unknown capsule type by an active node when such service code is no longer stored in local cache. All these nodes communicate between each other to have a global view of the network.

In order to better structure our architecture, the notion of Active Domain (AD) has been designed. In our definition, an AD consists of a control node and one or more active nodes. Each control node has autonomy in its AD; it manages and controls the active nodes located in its AD. Each control node updates the routing table, which corresponds to the active nodes of its AD, according to the deployed active services. It controls the deployment of services in the active network identifying appropriate active nodes that match the service requirement and are suitable for executing these services.

In most active networks design, the active nodes directory is centralized. This study proposes a novel approach based on decentralized system to get node information. A Peer-to-Peer system has been designed to achieve this functionality.

4.2 The P2P Layer

The P2P system is used as a Resource Discovery protocol to get network resources from active nodes in order to determine the active network topology. Due to the lightweight features of P2P systems, these systems are suitable for disseminating node specific capabilities to the service deployment modules in all active nodes. As it is difficult to distribute all the useful node information throughout the networks or the available resources that may vary rapidly, our proposal gathers only relevant information for service deployment. We followed the service requirements presented in [8] to guide our selection: topological requirements dealing with information about location, and resource requirements dealing with capabilities and availability of network resources. As in [3], our P2P system locates specific nodes and obtains information about their properties, and can be divided into the following tasks:

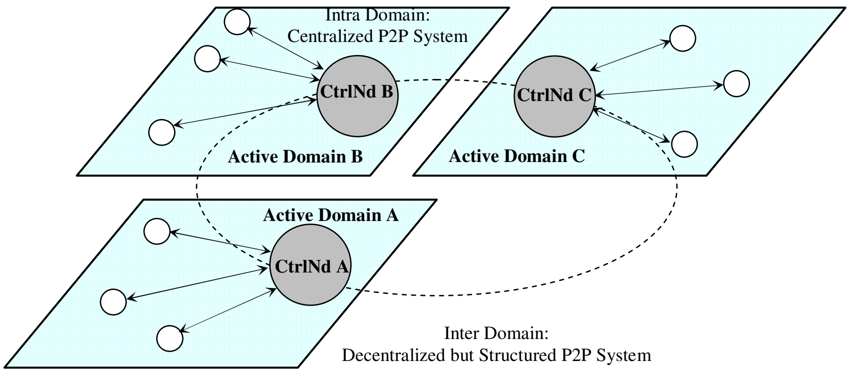


Fig. 3. P2P Scheme of Intra Domain / Inter Domain Nodes

- Monitoring local resources: Each active node is responsible for its local available resources.
- Advertising local resources: Each active node distributes information about its local available resources. This task may be done following either a centralized model among active nodes of the same active domain, or a decentralized model among control nodes of different active domains.
- Gathering information: Control node stores information about the active nodes of its active domain and information advertised from other control nodes.
- Propagating information: All information cached from other control nodes is also propagated throughout the active domains. That is, when a control node receives a state advertisement from a neighbor control node, it sends this information to all its directly attached neighbors.

To allow the Resource Discovery task, we propose a solution based on a hybrid P2P system. Locally, it consists of "small" centralized P2P systems, related to administrative domains, being controlled by one control active node acting as a super-peer. To allow a scalable active network, these super-peers are interconnected implementing a decentralized but structured P2P system (Fig 3).

5 Implementation

In our current prototype, the control node at the active network layer is implemented in Java and the resource discovery at the P2P layer in C++. Communication between them is possible through RPC over UDP. Both run on Linux, OpenBSD, and FreeBSD.

5.1 The Active Network Implementation

Active Network Transport System 2 (ANTS 2) with some extensions based on [7] has been used. The extended ANTS 2 is an active packet mechanism where capsules carry references to the code. Thus, using active packets, at the network level service deployment can be classified as a distributed/in-band approach [2]. At the node level, since capsules carry only the reference instead of the code, our extended ANTS 2 uses a centralized/out-of-band approach. That is, as a capsule arrives in a node, if the code is not locally cached, it is retrieved from a central code server.

As in [7], the modified version of ANTS 2 provides some APIs corresponding to the component of control (control node application) added to the platform, through which different, useful and short processing routines can be implemented (e.g. addService(), removeService(), viewService(), setControl(), getControl()…).

5.2 The P2P Implementation

The hybrid P2P system is chosen because of its capacity to deal with several types of nodes and situation. Therefore, it is suitable to our Controlled Active Network Layer. For example, some nodes with high bandwidth, CPU and memory are made control nodes and are responsible for managing the connection of the other nodes in a centralized way. Only these control nodes form a DHT. The Chord [13] system was chosen "to structure the decentralized system" due to its features of lookup performance.

In the centralized system, attributes of active nodes of an AD *A* are centrally stored in a table (LocalAttributesTable) of the control node *A*. Thus, it is enabled to provide the ability to search and choose suitable active nodes for service deployment on its active domain by executing a local search.

In the decentralized but structured system, a copy of active node attributes is stored in a table (ExternalAttributesTable) of other control nodes. This allows the communication among ADs and consequently, service deployment in active nodes of different ADs. As this storage employs a structured solution of new decentralized P2P system classes presented in section 3, each control node deals with small tables containing information of only few active nodes of other ADs.

Like [5], our proposed decentralized system is subdivided into three sub layers: Chord Lookup, DHT Storage and Resource Discovery Application. The main idea is to use a layer that translates high-level names into Chord identifiers (or keys). Thus Chord is only used as a lookup service. By layering additional features on top of a core lookup service, the whole system gain robustness and scalability. The functionality of each sub layer in our proposed P2P Layer are described here:.

Chord Lookup: this sub layer only provides mapping functionalities. As defined in [5], to implement these mapping functionalities, this sub layer maintains a routing table (Finger Table) at each control node with information about O(log N) other control nodes, where N represents the number of control nodes. It is not responsible for storing the binding key/attributes; instead it provides to the upper sub layer an efficient way of looking up the control node responsible for its storage.

DHT Storage: placed over the Chord sub layer, the DHT Storage uses the SHA-1 hash function to produce identifiers on a circular space. According to our proposal, each control node has a unique ID (controlnodeID) obtained by hashing its IP address. Each active node IP is also hashed to obtain a key k and the binding key/attributes is stored at the successor of k, that is, the controlnodeID that follows the k value. The DHT uses the Chord lookup service to locate the control node responsible for an active node.

Resource Discovery Application: this sub layer is responsible to provide interfaces of active node attributes to the control node application.

5.3 Resource Discovery Based on P2P

As shown in section 5.2, control nodes maintain two main tables to deal with resource discovery: the LocalAttributesTable and the ExternalAttributesTable.

The former is used by control nodes on intra domain service deployment. Once connected to a control node, each active node registers its IP and Attributes in the LocalAttributesTable.

The latter is used by control nodes on inter domain service deployment. Thus, after obtaining a key k by hashing the IP of the active node, the binding k/Attributes is stored in the ExternalAttributesTable of the control node which the controlnodeID succeeds immediately k.

The API provided by the Chord consists of five main functions, as shown in Fig 4. Thus, control nodes make available information of active nodes of its active domain calling insert (key, attributes). Active node information is updated calling update (key, attributes). Calling lookup (key), it is possible to efficiently get information in order to determine suitable active nodes to deploy the service. The final two APIs calls are functions for control nodes to join or leave the system.

API	Description
insert (key, attributes)	Insert a key/attributes binding at control nodes
lookup (key)	Return attributes associated with the key
update (key, newattributes)	Update a key/attributes binding at control nodes
join (activenodeID)	Join the system
leave (activenodeID)	Leave the system

Fig. 4. Description of the API

6 Future Works

The current prototype needs further refinements. Mainly the interactions between the P2P and the active network layers should be finalised in order to get a complete interoperability with many active network technologies, not only ANTS2. Some performance tests must be performed in order to know how efficient the proposed architecture is, in terms of resource advertisement, resource discovery based on the Chord lookup mechanism. It should also clarify how scalable the solution is, depending on the number of active nodes in the overall network, the number of active nodes belonging to one administrative domain. Works on an active network simulator like [17] may be developed/extended in order to achieve these goals.

It is well known that the active packets approach suffers from performance-related problems because of its great safety. The programmable networks approach has good performance because security issues are fewer than in the previous approach. However, the flexibility of such approaches is limited. In an effort to increase flexibility and to re-use interesting features on programmable networks, DAN and ANTS architectures have adopted a scheme where code is downloaded on demand and is cached for future use. The security issues are still complex and mandatory to be solved for the widespread deployment and use of active networks. As a future study, some remaining security issues must be considered.

7 Conclusion

In this paper, we presented a layered network architecture in order to improve the resource discovery, helping the service deployment process for active networks. Since the service deployment process is often closely coupled with the underlying active network platforms, our solution aims to provide an independent resource discovery mechanism, usable on various active nodes platforms, which help the module in charge of the service deployment to find the suitable active nodes depending on the service requirements and constraints. This resource discovery mechanism is based on the concept of P2P systems, which enable to disseminate active node information throughout the network without suffering from scalability issues. A P2P layer is then implemented on top of the active network layer. Moreover, by adding a control plane over the active plane, the service deployment is more efficient and reliable. Once the control node identifies a suitable set of active nodes using the P2P layer, a service code installation load is triggered in those nodes. The active applications code could then be automatically and dynamically deployed to those specific active network nodes in a single transfer.

References

1. Seo-Kyu Song, Stephen Shannon, Michael Hicks, and Scott Nettles. Evolution in Action: Using Active Networking to Evolve Network Support for Mobility. *IWAN'02*, Dec.2002.
2. M. Bossardt, T. Egawa, H. Otsuki, and B. Plattner. Integrated Service Deployment for Active Networks. IWAN'02, Dec 2002.
3. R. Keller. Dissemination of Application-Specific Information Using the OSPF Routing Protocol. TIK Report Nr. 181, TIK, ETH Zurich, November 2003.
4. R. Keller, "Self-Configuring Services for Extensible Networks – A Routing-Integrated System", Thesis of Swiss Federal Institute of Technology Zürich, 2004.
5. F. Dabek, M.F.Kaashoek, D. Karger, R. Morris, I. Stoica. Wide-area Cooperative Storage with CFS, *In Proc of the 18th ACM Symposium on Operating Systems Principles, 2001.*
6. L.Qin, C. Pei, C. Edith, L. Kai and S. Scott. "Search and Replication in Unstructured Peer-to-Peer Networks", *In International Conference on Supercomputing,* USA, 2002.
7. Rima Kilany, Ahmed Serhrouchni, Michel Riguidel, Dany Zebiane, "A Control Architecture for Active Networks*",* IEEE, International Conference on Software, Telecommunications and Computer Networks, SoftCom2001, Oct. 2001 Croatia
8. M. Bossardt, A. Mühlemann, R. Zürcher, and B. Plattner. Pattern Based Service Deployment for Active Networks, *In Proceedings of Second International Workshop Active Network Technologies and Applications* (ANTA 2003), Osaka, Japan, May, 2003.
9. M. Ripeanu, « Peer-to-Peer Architecture Case Study: Gnutella Network », Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01)
10. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. "Freenet: A distributed anonymous information storage and retrieval system." In *H. Federrath, editor, Designing Privacy Enhancing Technologies*, volume 2009 of LNCS, pages 46--66, 2000.
11. Tennenhouse, Smith, Sincoskie, Wetherall, Minden. "A Survey of Active Network Research". In *IEEE Communications Magazine*, Vol. 35, No. 1, pp80-86. January 1997.
12. Rowstron, A., and Druschel, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware* (Nov 2001), ACM, pp. 329–350.
13. Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc.Of SIGCOMM* (Aug2001), ACM, pp. 149–160.
14. Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. A scalable content-addressable network. In *Proc. of SIGCOMM* (Aug 2001), ACM, pp. 161–172.

15. Zhao, B. Y., Kubiatowicz, J. D., and Joseph, A. D. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. CSD-01-1141, U. C. Berkeley, Apr 2001.
16. Rima Kilany, Ahmed Serhrouchni. Using Distributed Component Model for Active Service Deployment, IEEE Symposium on Computers and Communications ISCC'2002, 1-4 July 2002, Taormina, Italy.
17. R. Kilany, A. Serhrouchni. ANS-2: An active general purpose simulator. On SSGRR'2003, July 28 - August 3, 2003, L'Aquila, Italy.

Mobile Agents for Testing Web Services in Next Generation Networks[*]

Abdelghani Benharref, Roch Glitho, and Rachida Dssouli

Concordia University, 1455 De Maisonneuve Bd West
Montreal, Quebec, H3G 1M8, Canada
abdel@ece.concordia.ca
{glitho, dssouli}@ciise.concordia.ca

Abstract. Web services are emerging as the standard paradigm for program-to-program interactions on Internet. They are gaining more and more momentum and are expected to play a key role in Next Generation Networks. For these reasons, their management is becoming critical. Fault management is one of the most challenging areas of network and service management. It includes fault detection, isolation and repair. This paper proposes a Web service-based architecture for detecting faults in Web services. The architecture is rooted in passive testing. Passive testing consists of observing and analyzing messages the component under test exchanges with its environment. The passive testers of our architecture are designed as Web services. The components of the architecture are presented and the potential interfacing technologies are evaluated. This evaluation has led us to the selection of mobile agents.

1 Introduction

Web services offer a set of mechanisms for program-to-program interactions over the Internet [1]. Managing them is critical because they are being used actually in a wide range of applications. Their use in telecommunications networks is being standardized by the open mobile alliance (OMA) [2]. Digital imaging is another application where their use is being studied [3].

Fault, configuration, accounting, performance and security are the main functional areas of network and service management. This decomposition has been enhanced in recent years to suit the needs of 3G systems. Fault management includes fault detection, localization, isolation, and repair. Detecting faults in networks and systems is no easy task. Widely used mechanisms include testing ([5],[6]) and alarm filtering and correlation ([7],[8]).

In testing, fault detection is usually based on previously generated test cases that are applied to the network or service under test. This procedure is known as active testing. It is carried out either before the deployment of the network and services, or when a problem is reported.

Another type of testing is passive testing. It can be carried out during development and also after deployment. In passive testing, messages sent and received by the

[*] A very early version of this paper was presented as a poster at IEEE IM05. This paper and the poster are substantially different.

component under test are observed, and the correct functioning of the component is checked. The observation is done by entities known as observers. Many models have been used for observers but most of the published works on passive testing are on control part of systems and use the FSM model [6].

Passive testing can complement active testing because it can aid in detecting faults that have not been detected before deployment. Furthermore, in many cases, it is a better alternative to active testing when problems are reported. It enables fault detection without subjecting components to test cases. Test cases consume resources and may even imply taking components off-line. However, since the test cases are appropriately generated, high fault coverage is guaranteed in active testing.

The management of Web services is highly platform-dependent today. Each platform comes with its own testing tools and these are in most cases active testers. This makes testing a vendor-dependent activity. It also makes testing of already deployed web services an activity where third parties are not allowed.

An open and multi-player testing environment is of paramount importance for the successful deployment of Web services. This will enable third parties including Web service requesters and third party certification entities to verify the correctness of the features claimed by Web service providers.

This paper proposes a novel architecture for detecting faults in Web services. The architecture is rooted in passive testing. The observers are designed as Web services. This makes them platform independent and opens the door to testing of deployed Web services by third parties. The potential is demonstrated by a case study where we test a deployed conferencing Web service. The overload and limitations are also discussed.

In the next section, we discuss related work. Section 3 introduces the architecture. After that, the case study is discussed. In the conclusion, we provide a summary and discuss the items for further work.

2 Related Work

Due to the recent introduction of web services concepts, most research in this area focus on the development and deployment of web services. Management of web services, and in particular fault management, is not yet a well-defined area. Researchers are still trying to enumerate the requirements and define the approaches.

Approaches developed for management of web services can be divided into two main groups: those based on active testing and those requiring the web service to support a management interface.

In the category of architectures that should provide a management interface, the WWW Consortium presents some of the requirements that web services management architecture should enable for the sake of management [9]. Similarly, The OASIS project proposes to manage web services and resources through manageability interfaces [10]. An approach where the web service provides a specific interface for management is presented in [11].

In the category of active testers, Parasoft [12] provides a set of tools (SOAPTest, .TEST, WebKing) to assist during the life cycle of a web service.

All the solutions presented in this section assume that a web service will participate in its management by providing specific interfaces or are based on active testers. The architecture becomes more complex if it has to support management aspects in addition to its basic operations. The performance of the web service is also degraded due to these additional features. Moreover, the developers of the web service have to implement also the needed interfaces and APIs to support management. When the tester is an active tester, appropriate test cases have to be carefully generated, executed, and their results analyzed.

As will be discussed all along this paper, our proposition for fault management is based on passive testing. Most of existing works on passive testing uses FSM. This model has been specially used for fault management in networks ([6]).

The observer can be either on-line or off-line: an on-line observer checks in real time the exchanged input/output while an off-line observer uses the log files generated by the component itself. Unlike the off-line observer, an on-line observer has the advantage of detecting faults when they occur but has the disadvantage of being more complex and resource demanding if a complicated model is used. In this paper, we focus on on-line observers due to their capability of detecting errors somehow quickly.

3 Components and Interfaces

The main components of our fault detection architecture are the observers. Since we are dealing with web services, we propose to design each observer itself as a web service. Two main reasons justify this decision. First of all, the observer must be available for service providers, their clients and third parties depending on who is interested in observation. A service provider can use the observer to make sure that the web service she/he is offering is operating correctly. The client can make use of the observer in order to verify that the service she/he is using is operating as agreed on with the service provider. Furthermore, a third party mandated by the client and/or the server, can invoke the observer to provide an independent certification stating if the web service is correct or not. In addition, no additional skills are required since invoking the observer follows almost the same steps as when invoking another web service.

3.1 Requirements

In addition to the on-line detection capability, the passive property of the tester is the major design key in our architecture. In fact, the following points have been considered while designing our tester:

• Online detection: the tester should be able to detect faults as soon as they are generated by the observed service. This implies that the observer(s) must receive, using appropriate mechanisms, all the messages generated by the client and the service.

• Minimize the induced load: even if all testing activities will introduce some extra load, the idea here is to keep this load as low as possible. This idea is of prime importance since web services will be deployed in wireless environment where the network load is a critical issue. Interestingly enough, the passive tester will never generate messages just for testing except when reporting the result of the observation.

• Transparency: in a completely distributed architecture, neither the service nor the client will notice that testing activities are taking place. The communications mechanisms used for traces collection should have no impact on the delivery of exchanged messages between clients and services.

3.2 Procedure

The testing procedure within our architecture can be performed following the steps listed hereafter and illustrated in Fig. 1.

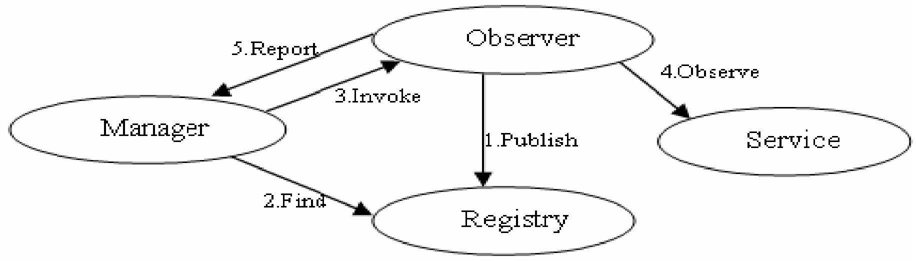


Fig. 1. Observer Invocation

Step **1.** Publish and **2.** Find operations are the same as defined in the web services architecture [1]. In Fig. 1, dynamic publication in UDDI registry is considered.

3. Invoke: The observer is invoked by a manager. The manager can be at the client side (but not necessarily the client application), the manager of the web service or even an authorized third party. In all cases, the manager should provide specific information when invoking the observer. This information includes the location of the service to be observed, its specification, when to start and when to stop observation, etc.

Once the observer is invoked, it parses the FSM machine of the service to be observed using the information provided as parameters then starts analyzing the observed traces and comparing them to the expected traces. The FSM model can be derived from the original specification of the service to be observed. It can be described in a simple XML document.

4. Observe: This is the main action in the monitoring of web services. The observation can start for new service invocations or for service invocations already in progress. The fault detection procedure is the same but the homing procedure is longer for the latter case [6].

(a) New service invocation: since the observer will start the observation when the service starts interactions with the requester, the homing sequence is empty.

(b) Invocations in progress: unless the set of missed input/output pairs are handed to the observer at invocation, the homing procedure is necessary to deduce the homing sequence that brings the observer to a state equivalent to the one the service specification is in. After successful homing procedure, the fault detection procedure starts.

5. Report: When the observer stops observation (fault found or observation period expired), it returns the result to the manager. Based on these results, fault location, isolation and correction processes are initiated if a fault occurred.

3.3 Considerations on the Communication Between an Observer and Other Entities

One of the design keys to be inspected is how to provide the observer with all exchanged input/output sequences. The flexibility of the architecture allows the manager to choose, from a variety of possible communication mechanisms. These possible mechanisms include the instrumentation of the client and/or the service, SNMP agents/managers, dispatchers and mobile agents.

Client/Service instrumentation. An obvious solution is based on a high participation of the client and/or the service. It supposes that the source code of the client and/or the service is available and can be instrumented to forward messages to the observer.

SNMP. A SNMP agent accessing a complete MIB sends all the entries to the observer, which plays the role of an SNMP manager in this case. A complete MIB, in this case, is a MIB that holds all the exchanged information between a client and a service.

Dispatcher. An interesting solution is to design a special dispatcher. The dispatcher gets, all exchanged messages and forwards them to other entities. This mechanism is completely transparent to the service/client and does not require them to invoke any interface from the observer. Invocations are performed by the dispatcher itself.

Mobile agent. In the mechanisms presented above, the observation capabilities are all put in the observer side, and the problem to solve was to collect and move traces to the observer. A completely different approach is to move the observer itself to the service/client side instead of the traces. This is made possible by using mobile agent technologies [15]. The observer, when invoked, can generate and send a mobile agent to the calling platform

The following table summarizes the requirements of each communication mechanism and illustrates the level of participation of client/service for each mechanism:

Table 1. Communication mechanisms characterestics

Mechanism	Client/Service participation	Changes required to client/service	Network Load	Time Delay
Client Instrumentation	High	High	High or medium	None
SNMP	Low or None	Low or None	High	None
Dispatcher	None	None	High or medium	Low
Mobile Agent	Low or None	Low or None	Low	None

More technical information and measurement of the network load introduced by each of these mechanisms are discussed in section 4.3.

4 Implementation: Call Control

To illustrate the applicability of our architecture, a web service [16] exposing some of the functionalities of Parlay/Call Control [17] has been observed using FSM models and the algorithms presented in [6]. This web service provides mainly the conferencing parts of the Parlay specification: create conference and sub conferences, add, move and remove users, end conferences and sub conferences.

4.1 Example of Testing Platform

All the communication mechanisms presented in section 3.3 have been implemented and their performance evaluated. Except for the network load, all the mechanisms give similar results with regards to the effectiveness (detection capabilities) of the architecture. Due to the flexibility of the graphical interface of the instrumented client in terms of selection of scenarios and observation of results, we present in this section the testing platform of the client instrumentation.

The client application is instrumented in such a way to forward all exchanged messages with the web service to the observer and process its decisions. The client application is also responsible of adapting these messages for the observer. The components of the testing platform are depicted in Fig.2.

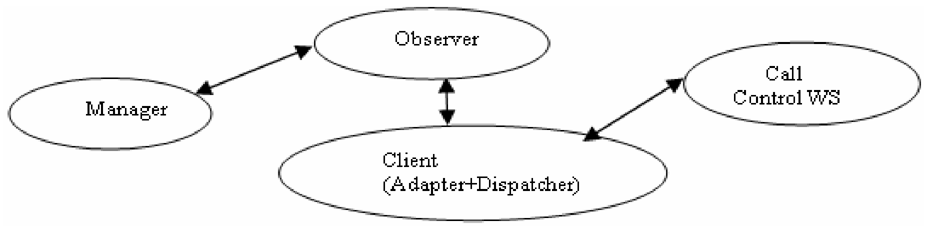


Fig. 2. Client instrumentation

- Manager: the entity interested in the observation of the Call Control web service. In our case study, it is the client application itself.
- Call Control WS: the web service (to be observed) exposing the conferencing capabilities, deployed on Ericsson Application Server.
- Observer: the tester designed as a web service, implementing FSM algorithms in [6] for the homing procedure and fault detection. The observer is deployed on BEA WebLogic platform [18].
- Client: the application that invokes the web service. It also adapts (adapter) and forwards (dispatcher) messages to the observer.

During the invocation, the manager provides the XML document describing the FSM model of the Call Control web service to the observer. The invocation is performed using the "initObserver" interface. The observer then parses the XML document and uses a special data structure to store the FSM. If the parsing succeeds, the observer is then ready to process input and output events. In our architecture, the input stands for messages from the client to the service and the output for messages sent back from the service to the client. The client application sends input and output to the observer using "sendInputToObserver" and "sendOutputToObserver" interfaces. The observer checks the validity of each pair (input/output) and returns a decision to the client.

4.2 Results and Analysis

Effectiveness of the Architecture. The graphical interface of the client application allows the user to execute a wide variety of valid or invalid scenarios, and checks the observer's decision for each executed scenario. Table 2 reports some of the scenarios that have been executed with their observed verdicts.

The "initiateConf" fails twice during observation. When it's invoked with an empty list of initial users, no response is received. Also, giving a conference type different than "dial out" has no effect and a dial out conference is initiated in all cases. The first fault cannot be detected by the FSM-based observer since no output is received even when waiting for a long time. The second fault, in the contrary, is detected. In the scenarios when the observer detects an INPUT fault, the FSM machine was in the "Idle" state. At this state, the client can only initiate a conference. An INPUT fault states that the client requests an operation from an invalid state. The web service passes all tests when an INPUT fault was observed. After observing the Call Control web service, it seems that the web service, and the Parlay gateway behind, is more or less well tested (with regards to the executed scenarios) except for the two cases for which the web service fails.

Table 2. Some executed scenarios and verdicts

Target	Scenario (Valid/Invalid)	Verdict	Comments
initiateConf	Valid (With initial users)	Pass	
initiateConf	Valid (List of initial users empty)	Fail	No response, no decision received from the observer
initiateConf	Invalid (wrong conference type dial out/dial in)	Fail	A dial out conference is initiated, a fault is detected
AddUser	Valid	Pass	
AddUser	Invalid (add a user before initiating a conference)	Pass	The observer detects an INPUT fault
moveUser	Valid	Pass	
moveUser	Invalid (move a user that does not exist)	Pass	
EndConf	Valid	Pass	
EndConf	Invalid (end a non-existent conference)	Pass	The observer detects an INPUT fault

For each input sent to the service or an output received from it, a message is generated to the observer by invoking respectively "sendInputToObserver" and "sendOutputToObserver". The traffic observed on the network infrastructure during an observation is almost twice the traffic without observation. The two interfaces have one parameter each (the event). Due to its structure, a SOAP message carrying one parameter has almost the same size (from a complexity point of view) as a SOAP message carrying two parameters. Merging the two interfaces into one ("sendInputOutputToObserver") will reduce the overload by almost 50%.

Quantitative Evaluation of Communication Mechanisms. The communication mechanisms presented above are easily configurable. To instrument the client (or the service), 4 basic java statements, in a try/catch block have to be added to the code to initialize the observer. Then, for each request to the service or response received from the service, 2 java statements, also in a try/catch bloc, have to be appended to the source code. The network load in the client instrumentation depends on which interfaces are invoked. Since the size of a SOAP message is around 2 kbytes, invoking the "sendInputOutputToObserver" generates half the network load generated by the two individual interfaces.

The TCP-dispatcher requires the modification of the WSDL document of the service to be observed. The IP address and the port number in this document should be replaced by the IP address and port number on which the dispatcher is listening. Here again, the size of a SOAP message is around 2Kbytes. The overall load depends on which interfaces are used: "sendInputOutputToObserver" or "sendInputToObserver"/ "sendOutputToObserver".

The observation can make use of the SNMP if a complete MIB and a SNMP agent are available. The agent generates trap from the MIB and send them to the observer. The agent requires the path to the MIB, the IP address and port number on which the observer is listening. Each entry in the MIB (input or output) is sent in a separate packet. The size of each packet is between 150 and 200 bytes.

Designing the observer as a mobile agent requires a flexible mobile agent platform. The Java Agent Development Framework (Jade) [19] has been used. In addition to the open source feature, its flexibility, ease of configuration and use justify its utilization in the design of our mobile observer. The platform is a set of libraries that can be downloaded and configured offline. The size of these libraries is around 8 Mbytes, The size of the mobile observer is around 620Kbytes. Work is still taking place to reduce the size of the mobile agent by optimizing the required libraries.

Comparison and Selection. The network load is a critical evaluation criterion in wireless environment where web services will be deployed. Fig. 3 illustrates measures of the network load introduced by the observation using the communication mechanisms presented above. Even with more or less big size of the mobile agent, the mobile agent approach presents the lowest network overload. This is due to the fact that the observation traffic to transit over the network is constant and independent from the number of times the service is invoked.

If the number of request/response is less than 2000 request/response, only the SNMP approach gives less or similar load as the mobile agent. For more than 2000 request/response, the load generated by the SNMP approach continues to grow while the load introduced by the mobile agent mechanism remains constant. Since the number of request/response for a service will be higher than 2000, we conclude that mobile agents are the most suitable mechanism for communication between different components of the architecture.

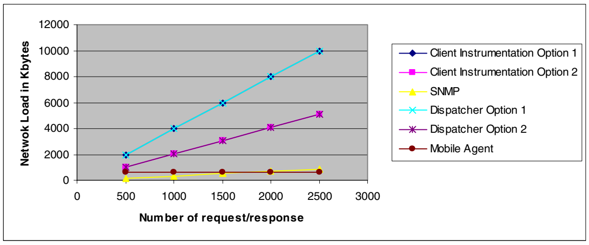


Fig. 3. Network load measures for communication mechanisms

5 Conclusion and Future Work

As web services technology emerges, its management becomes an important issue. Management includes aspects from fault management to security management. In this paper, we presented an architecture for fault detection of web services through web services based on passive testing and illustrated its applicability with a case study. The main contribution concerns the design of an observer that can be invoked by interested parties when developed and published as a web service.

The models that can be used within an observer should be further studied to identify the most appropriate models for web services or for a category of web services. We will investigate a set of potential models; especially the Extended Finite State Machine (EFSM) model as a base model for our observers to deal with data flows in web services interactions. This is motivated by the fact that data flows are very important in XML-based distributed computing. The homing procedure for an on-line observer, when using an EFSM model, is more complicated since variables have to be initialized correctly before fault detection starts. We will study and propose a set of algorithms and techniques to simplify an EFSM-based observer. Performance evaluation and measurement of timing properties is another interesting aspect in web services industry. A model should be developed to allow performance monitoring using web services-based observers.

This architecture will be improved in future work to support the remaining steps in fault management: location, isolation and fault repair. The number and location of the points of observation affect significantly the detection capability of the testing system. If the final service is a composition of many services, it might be more interesting to consider a network of observers, an observer for each service rather than an observer for the final composed service. Cooperation of all observers can generate pertinent information for service management. The scalability is another issue that will be studied in our future work.

Acknowledgements. This work has been partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC) and Ericsson Research.

References

1. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/
2. http://openmobilealliance.org
3. http://www.i3a.org/i_cpxe.html

4. Universal Mobile Telecommunication System (UMTS); Telecommunication management; Principles and high level requirements (3GPP TS 32.101 version 6.1.0 Release 6).
5. C. Kaner et H.Q. Nguen, Testing Computer Software.John Wiley & Sons, April 1999.
6. D. Lee et al. Passive Testing and Applications to Network Management Proceedings of IEEE International Conference on Network Protocols, pages 113-122, October 1997.
7. G. Jakobson et M. Weismann, Alarm Filtering. IEEE Network, Volume 7, Issue 6, November 1993.
8. R.D. Grdner et D.A. Harle, Methods and Systems for Alarm Correlation. GLOBECOM'96, Pages 136-140, Vol 1, November 1996.
9. Web Services Endpoint Management Architecture Requirements Draft, http://www.w3.org
10. http://www.oasis-open.org
11. J. A. Farrel and H. Kreger, Web Services Management Approaches. IBM Systems Journal, 41(2), 2002.
12. http://www.parasoft.com
13. R.E. Miller, Passive Testing of Networks using a CFSM Specification IEEE International Performance Computing and Communications Conference, pages 111-116, February 1998.
14. A.R. Cavalli et al., New Approaches for Passive Testing using an Extended Finite State Machine Specification. Information and Software Technology. Vol. 45, No. 15. September 2003.
15. A. Fuggetta et al., Understanding Code Mobility. IEEE Transaction on Software Engineering, Vol.24, No 5, 1998.
16. K. Hassan et al., Web Services for Call Control in 3G Networks: A Parlay Based Implementation. 9th IEEE International Conference on Network Protocols. Pages 122-131, November 2002.
17. Parlay 4.1 Specification, http://www.parlay.org
18. BEA WebLogic plateform, http://www.bea.com
19. http://jade.tilab.com

A Secure Protocol for Mobile Agents in Hostile Environment

Abdelhamid Ouardani, Samuel Pierre, and Hanifa Boucheneb

Mobile Computing and Networking Research Laboratory, (LARIM),
Department of Computer Engineering, École Polytechnique de Montréal,
P.O. Box 6079, Station Centre-ville, Montréal, Québec, Canada H3C 3A7
Phone: (514) 340-4711 ext. 4685, Fax: (514) 340-5159
{abdelhamid.Ouardani, samuel.pierre,
hanifa.boucheneb}@polymtl.ca

Abstract. Mobile agent systems present a good solution to implement applications that, for a long time, have used client/server architecture. However, this technology is confronted with security threats from agents and hosts. This paper presents a protocol which protects mobile agents from malicious hosts. This protocol is sufficiently robust so that it is durable and fault tolerant.

1 Introduction

This paper proposes a protocol based on sedentary and cooperating agents to protect mobile agents against malicious hosts. This protocol combines four concepts: the cooperation between mobile and sedentary agents, the reference execution within a trusted platform (reliable platforms which run our cooperating sedentary agents), the cryptography and the digital signature to make inter-agent communication secure, and the time-limited execution (Timeout). A dynamic approach which makes use of a time-limit to make it possible to detect mobile agent's code re-execution, was selected. The main contributions of this paper are: the design of an approach to estimate the time a mobile agent takes to carry out a task on a platform in order to detect a code re-execution attack; the design of some mechanisms which allow the protocol to recover after service or permanent code modification has been denied and to become fault tolerant in case the trusted platform breaks down.

The remainder of the paper is organized as follows. Section 2 presents security problems and protection approaches of mobile agents. Section 3 presents the foundations and specifies the proposed protocol. Section 4 gives some implementation details and analyzes some experimental results. Section 5 concludes the paper.

2 Background and Related Work

This paper studies the protection of mobile agents in hostile environment. This issue is problematic given the fact that the platform is the environment required to execute these agents. The platforms hold all the elements necessary for agents' execution. Some researchers treat mobile agent security by studying attacks [10][11][13].They analyzed the attacks, presented the relationships between them [1][3] and identified their level of importance [1]. The common conclusion stated eavesdropping as the most dangerous attack, as their occurrence can lead to more dangerous attacks, such data modification.

Security threats can be classified into four categories [13]: integrity attacks, availability attacks, confidentiality attacks, and authentication attacks. *Integrity attacks* include any attack which modifies or interferes with the code, the state or the data of the agent. *Availability attacks* occur when an authorized agent cannot access objects or resources to which it has the right. *Confidentiality attacks* happen when the components of the agent are accessed illegally by a malicious platform. *Authentication attacks* occur when a mobile agent cannot identify and validate the current platform to which it is assigned or certified.

Existing methods differ in their capacity to immunize, protect or detect potential attacks. At this stage, it is interesting to specify that the moment of detection occurs either immediately after the attack was made or even later. Moreover, even if the protection is incomplete, it is still necessary as it limits the damage and thus minimizes the consequences. Some researchers have proposed approaches based on the calculation of cryptographic functions [3], which aim to prohibit decrypting the agent's code. One problem with this approach is that it is impossible to find cryptographic functions for all of the problems.

A mobile agent system, called *Ajanta* [12], offers mobile agents three mechanisms of protection. For the first one, the agent's author can declare a part of the agent's state to be read only (*ReadOnlyContainer*). The second makes it possible for the mobile agent to create append only logs *(AppendOnlyContainer*); however, this approach is not appropriate when two platforms are working in collaboration. The third is selective revelation which enables the platform of origin to appropriate certain information from the agent to well-defined platforms (*TargetedState*). In other words, only part of the agent's data will be visible to the platform for which it is intended. Others have proposed approaches which protect the agent's state or behavior by comparing it to one which has been used as a reference [5][7][15][4]. These approaches detect attacks even if some platforms are working in collaboration. [15].

A secure protocol has been proposed in [1] and makes use of a sedentary agent which carries the mobile agent's critical code (the part of the code that consists of its objective function, for example). This cooperating agent is carried out within a trusted platform to insure its itinerary. All of its results are compared to those previously received from the mobile agents from each platform visited to detect any wrong execution by the agent. Then, it sends the originating platform a report identifying all of the malicious hosts which attacked the mobile agent. This approach also uses two time counters [6] to detect mobile agent re-execution and denial of service. To insure confidential communication between the mobile agent and its cooperating agent, this approach uses signed data sent through encrypted messages based on a public/private key system. It can detect attacks like code re-execution, denial of service, wrong execution, itinerary modification, and eavesdropping. The drawback to this approach is the estimate made by the time counter. It uses another agent to simulate the mobile agent. Thus, the agent's owner can deduce the execution time required on a platform. However, this mechanism might make mistakes depending on platform quality (a fast platform can re-execute the agent without being detected, while a slow platform can be considered as suspect while it is honest). Also, a permanent modification of the mobile agent can effect the agent as it continues uselessly its migration. Moreover all future hosts will be considered as being malicious even if they don't.

3 Secure Protocol for Mobile Agents in Hostile Environment

The proposal makes use of three different entities: a mobile agent (*MA*) and two sedentary agents (*SA* and *SAR*). The *MA* migrates between a set of hosts (P_i, which could either be malicious or honest) forming its itinerary $Id(P_i),Prev_i$ and $Next_i$ the P_i*'s* identity, P_{i-1}'s identity and P_{i+1}'s identity, respectively. *Time* represents the *MA's* time of execution and P_i , T_{ai} and T_{id} respectively designate the inter-arrival on platform T of messages *Arrival()-Input()* and *Input()-Departure().* Note *Timeout* and *IAS* as the estimated time counters after which a code re-execution and denial of service will be detected. Also note *TimeoutSA* as the maximum amount of time required by *SA* to cooperate with an *MA* and update the *SAR*. Note $E_{Pi}(X)$ as the encryption of the data X using P_i's public key, $D_{Pi}(X)$ for the decryption of X using P_i's private key and $SIG_{Pi}(X)$ for the digital signature of X using P_i.

Assuming the existence of two trusted platforms within the protocol: a primary one called *T* where the *SA* is executed, and a second platform for recovery called *TR* on which the *SAR* is carried out. The mobile agent code is divided into two parts: the sensitive one is called the critical code and the non-critical code. The critical code is also retained by the *SA*. Assuming that the messages exchanged between agents are carried through authentic channels, to insure communication, an asymmetrical cryptographic system [1] must be made available so it will be possible for these agents to get their private and public keys required for encryption as well as a one-way hashing function to make it possible to send signed data.

At the initial stage, the protocol is launched starting from the platform of origin which creates the *SAR*, the *SA* and the *MA*. It initializes the route recording parameters ($Prev_0$ with $Id(O)$ and $Next_0$ with $Id(P_1)$) for the agents. It also initializes parameters like *Timeout* time, *IAS* and *TimeoutSA* with the preliminary values chosen. Also, it provides the *MA* with a list of the known malicious platforms to be avoided, and charges a list of alternate platforms on the *SA*. It then sends the *SAR* and *SA* respectively towards the *TR* and *T* platforms. For the last step of this stage, it sends the *MA* towards the P_1 platform. The *MA* knows the identity of the *T* and *TR* platforms and those of the *SA* and *SAR* platforms.

During step *i*, the two cooperating agents (*MA* and *SA*) exchange three types of messages: *Arrival()*, *Input()* and *Departure()*. *Arrival()* contains the identity (*Idi*) of the current platform P_i; $SIG_{Pi}(Id_{AM})$ the identity of the *MA* signed by P_i; $Prev_i$ the identity of platform P_{i-1} (the preceding platform along the *MA*'s path), and *StateMA* the current state of the *MA*. *Input()* contains Id_i the identity of platform P_i, X the data input provided by P_i to the sensitive part of the *MA* code and $SIG_{Pi}(X)$ the signature of data X using the public key of P_i. Finally, the *Departure()* message contains the identity of platform P_i (Id_i), the results R obtained by the mobile agent, the $SIG_{Pi}(R)$ signature on R using the public key of platform P_i, the identity of the next platform (P_{i+1}:$Next_{Pi}$) and the state of the *MA* (*StateMA*). The whole protocol will be summarized later on Figure 2.

Once it arrives at P_i, *MA* sends an *Arrival()* message to the *SA* which is awaiting the *MA*'s messages. The *SA* verifies its authenticity by examining Id_{AM}. It also checks if the *MA* really migrated towards platform P_i as expected ($Next_{i-1}= Id_i$). If this is not the case, it recognizes an itinerary modification attack [2][1][14], and it then examines the *MA* to see if $Prev_i$ is equal to Id_{i-1} which ensures that P_{i-1} actually sent the *MA*. If the preceding equations are satisfactory, then the sedentary agent arms the timer (*Timeout*) and waits to receive the remaining messages from the mobile agent. This timer is used to measure the execution time of the *MA* on platform P_i. If it expires, the *SA* will suspect a re-execution of the mobile agent. The *SA* also initializes a timer which will measure T_{ai} (inter-arrival of *Arrival()* and *Input()* messages coming from the *MA*). Before executing the critical code, the *MA* sends an *Input()* message to its cooperating agent. This message is encrypted with the public key for the *T* platform. In this message, the mobile agent provides the *SA* with the data required to execute the critical part of the code. The *SA* agent extracts the data signed by platform P_i with its private key, then, it stops the timer and evaluates the T_{ai}. Using this value, the *SA* calculates the right time T_{id} (inter-arrival of *Input()* and *Departure()* messages coming from the *MA*) before it receives the *Departure()* message. In fact, the cooperating agent (*SA*) uses a function *f* which implements an empirical law to deduce T_{id} from T_{ai} ($T_{id} = f(T_{ai})$). Then, the *SA* deduces the *MA*'s total execution time on current platform P_i using the formula bellow:

$$\boldsymbol{Time = g\,(T_{ai} + T_{id}) = g\,(T_{ai} + f\,(T_{ai}))} \qquad (1)$$

where *g* is a function which implements the relationship between the execution time of the *MA* and the sum of the inter-arrival time of the *Arrival()-Input()* and *Input()-Departure()* messages. This ***Time*** value is then corrected by a multiplying a coefficient for safety estimated during the experiments carried out by the protocol. Hence, one obtains the value of the new *Timeout* which takes into account the characteristics of platform P_i in terms of performance, the number of agents carried out and also the load on the communication channels between P_i and *T*. In other words, the cooperating agent readjusts the preliminary value of the *Timeout* already initialized to detect a possible attack on code re-execution.

Whereas the mobile agent continues its execution, the *SA* also begins to carry out all of the critical code, since it has received all of the necessary input. Once finished, the *MA* sends a *Departure()* message ($Id(P_i)$, *R*, $SIG_{Pi}(R)$, $Next_i$, *StateMA*). It encrypts it with the public key for platform *T*. When the *SA* receives this message, it decrypts it using its private key of platform *T*. It then extracts the results and checks the signature of platform P_i. The results are compared with those obtained by the *MA*. If they are not identical, a modification of code attack is detected. Finally, the *SA* records this information and waits until the mobile agent migrates towards the next platform.

As previously mentioned, the *SA* readjusts the *Timeout* of the protocol, which is armed as soon as it receives the *Input()* message. Therefore, if the *Timeout* expires before receiving *Departure()* message, then *SA* recognizes this platform as a re-execution attacker. Otherwise, it continues to wait for the *Departure()* message within another interval of time (*IAS*) [1]. If *Departure()* message still has not arrived, the *SA* will detect a denial of service.

Before starting another step, the cooperating agent makes a summary of the *Arrival()*, *Input()* and *Departure()* messages, produces and sends a new *UpdateSAR()*

message to the *SAR* to provide it with information about the latest execution of the *MA* in platform P_i. The *UpdateSAR()* message, whose the structure is shown in Figure 1, contains the identity of the cooperating agent encrypted with the private key of *T*, the identity of the *MA*, the identities of the previous ($Prev_i$), the current (Id_{Pi}) and the next ($Next_i$) platforms and both the input (*X*) and the results (*R*). It also contains a vector (*Attacks*) of the attacks detected within the current execution. Once it receives the *UpdateSAR()* message, the *SAR* records the data it contains, starts its *TimeoutSA* timer to detect a possible breakdown of the sedentary agent on platform *T*. *TimeoutSA* is estimated by the *SAR* at the beginning of the protocol using the same approach that the basic protocol [1] uses to estimate its *Timeout*. Indeed, before its migration towards the *T* platform, the *MA*'s cooperating agent initially sends its first *UpdateSAR()* message to the *SAR* which is already running on platform *TR*. Then, the *SAR* simulates an execution of the *MA* and sends another *UpdateSAR()* message to the *SAR*. Thus, one obtains an estimate of the inter-arrival time of the two *UpdateSAR()* messages between two executions. To this estimate, the *SAR* adds the value of the *IAS*, so, the protocol will ignore if the *SAR* suspects a breakdown of the *SA* while it is still running. The *SA* does not declare a denial of service to the *MA* as it is waiting for its messages.

Id_{MA}	$Prev_i$	Id_{Pi}	X : *Inputs*	R:Results	$Next_i$	$SIG_T(Id_{SA})$	*Attacks*

Fig. 1. Structure of the *UpdateSAR( )* message

During stage *i*, three different situations can occur: a fault on the *T* platform, a denial of service or a permanent code modification:

Fault on the T Platform: The *SAR* does not cooperate with the *MA* during its various migrations except during breakdown situations. It only interacts with the *SA*. In fact, the mechanism works as follows: the cooperating agent sends the *SAR* a summary of the three messages exchanged through an *UpdateSAR()* message. Once it receives this summary, the *SAR* starts a timer with the value of *TimeoutSA*. Thus, if the *SA* does not send any more messages and the timer expires, the *SAR* suspects a breakdown of the primary cooperating agent (*SA*). Consequently, it sends a *changeCo-operanting()* (a *hello)* message to replace the current cooperative agent of the *MA* to inform it of becoming the new cooperating agent. The *MA* starts by sending the three previous messages as described above, which precede the detection of a breakdown. The *changeCo-operating()* message contains the identity of *SAR* (Id_{ASR}) and the signature of platform *TR* on Id_{ASR} ($SIG_{TR}(Id_{ASR})$).

Denial of Service: This occurs when, for a given reason, the *MA* cannot continue to function. The protocol detects this situation using a timer *IAS* (additional waiting interval).

To avoid restarting the protocol from the beginning, the cooperating agent creates a copy of the updated *MA* with all of the information that the previous mobile agent had. It is for this purpose that the cooperating messages contain the *StateMA* parameter, thus making it possible for the new copy to start from where the previous copy was stopped functioning Then, the cooperating agent sends the copy to an alternative platform chosen from a previously determined list. Once on the new platform, the new *MA* can continue its mission.

Permanent Code Modification: This is suspected when the cooperating agent detects two consecutive attacks on a simple modification (on two platforms P_i and P_{i+1}). In this case, it will eliminate the mobile agent on platform P_{i+2} since it has already migrated from there. Then, it will send a valid clone towards the previous platform (P_{i+1}) and wait for the cooperation messages from the new *MA*. Once it receives the *Departure()* message, it will compare the results received with those obtained from the original *MA*. If they differ, it concludes that P_{i+1} is actually a malicious host and it is identified as such.

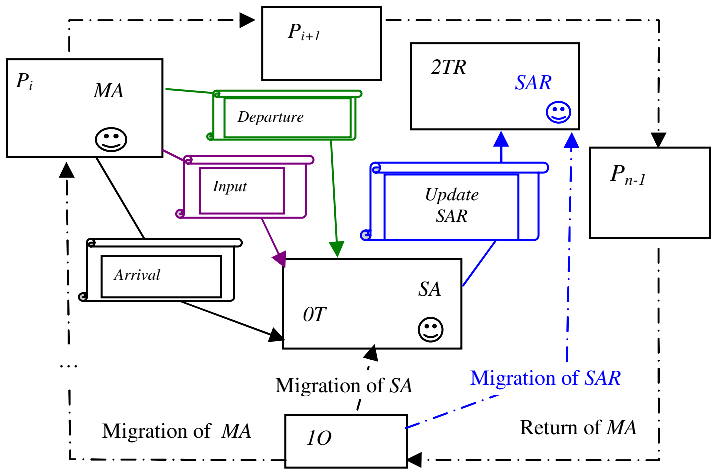


Fig. 2. Secure protocol based on sedentary agents

At the end of the protocol, the mobile agent returns to its platform of origin (*O*). The latter identifies which of the two sedentary agents cooperated with the *MA*. It requires a final report and a list of the malicious platforms. Figure 2 illustrates the whole protocol.

The detection of a code re-execution is based on a dynamic method of estimation of the *Timeout*. This approach does not leave a malicious platform an opportunity to perpetrate an attack on re-execution without being detected. Not only does it detect the denial of service, but it also makes it possible for the protocol to survive such an attack.

One could even indirectly consider it a measure of detection and protection. As for the attack on permanent code modification, the protocol allows for the recovery in case of the breakdown of one of its entities, which makes it possible to avoid several useless migrations. Also, the new protocol invalidates the need for an entity called the estimator agent since the protocol performs their required estimates. However, the proposal certainly produces some constraints, such as the addition of a second platform of confidence, which generally consists of a paying service, possibly increasing costs. However, the recovery procedures after the faults are detected are expensive in terms of time, memory resources and create traffic problems within the network. Indeed, they are necessary in order to generate a valid clone of the mobile agent if a denial of service occurs, to eliminate the *MA* and to generate its clone if a permanent code modification occurs, and finally to retransmit certain messages if an *SA* break-down occurs.

4 Implementation and Experimental Results

To implement the protocol, the Grasshopper platform [9] was used. This latter allow agents communication through remote invocation. The platform is organized as follows: all the participating entities are registered in a logical domain called a region. This entity contains a set of a logical subdomain: agencies, each of them can receive and create agents; it also contains special places where the visiting agents can be executed. These places provide services such as encryption and digital signature to make it possible for agents to exchange authentic messages. In this experiment, all the platforms were personal computers (P IV 1.6 GHz- RAM: 512MB) running Windows 2000, and connected to an Ethernet 100 Mbps.

When the *MA* runs on a P_i platform and before sending its first *Arrival()* message, it must initially activate the security context based on the provider IAIK_JCE3.03 [8]. Consequently, for all future platforms visited, it would thus be necessary for the *MA* to activate this context before starting its execution: this is the "cache" aspect. In addition, there is the so-called "proxy" image of the cooperating agent that the *MA* must build to locate its interlocutor. Thus, the *MA* could call upon the *arrival()* of *I_SA* method of implementing the *Arrival()* message. This is true for the first migration as, once on the platform P_{i+1}, the *MA* already knows the address of the proxy, therefore it does not need to look it up. Also, at the time of recovery after a denial of service, the *MA* is obliged to build the proxy before carrying out the cooperation. Consequently, during the experiments these two aspects were taken into account: no cache/no proxy (worst case), and no cache with proxy (best case).

4.1 Evaluation of Timeout

A series of experiments were conducted using both scenarios to find empirical functions to represent these relations: T_{id} = F (T_{ai}) and Time = G (T_{ai} + T_{id}).

1. *No Cache with Proxy (Best Case)*:

Initially, the results obtained compared to T_{id} = F (T_{ai}) measurements show a correlation between T_{ai} and T_{id}. A linear approximation leads to the following equation:

$$\mathbf{T_{id} = -0.2079 * T_{ai} + 307.53} \tag{2}$$

During the second trial, the results were obtained and are representative of the Time= g (T_{ai} + T_{id}). This latter is proportional to the sum of T_{ai} and T_{id}, due to the fact that, if the exchanged messages (represented by (T_{ai} + T_{id})) take awhile to arrive at their destination, it is due to the execution of *MA* (*Time*) took up some of their allotted time. The variation is mainly due to the fact that the time required by the *MA* to build the proxy as well as the time to start the *PlaceCrypto* (place of platform P_i) service is not considered in the $T_{ai}+T_{id}$ value. In fact, the evaluation of the latter ($T_{ai} + T_{id}$) starts only after the *Arrival()* message has been received. As a result, the searching proxy and starting the cryptographic services of the place procedures are not taken into account when it comes to measurement. On the other hand, to measure the value of the *Time* parameter all of the steps are considered. They are obtained through a linear approximation; the two parameters are related as bellow:

$$\boldsymbol{Time = 1.3049*(T_{ai} + T_{id}) - 62.218} \tag{3}$$

Thus, by combining relations (2) and (3), the ollowing relation is obtained:

$$\boldsymbol{Time = 1.033* T_{ai} + 401.29} \tag{4}$$

In fact, the execution time of *MA* on a platform P_i is estimated if the *MA* already has the proxy of the cooperating agent. The estimate is made according to the inter-arrival time of the first two (*Arrival()* and *Input()*) messages. In order to estimate a value for our *Timeout*, we did a series of measurements to compare the actual execution time of the *MA* with the estimate according to the relation (4). Indeed the latter (4) was introduced into the code of the sedentary agent to calculate *Timeout* and some improvement were conducted using coefficients. The comparison of the average deviation calculated to the actual time of execution showed that the standard deviation of the estimate, corrected by a 0.975 coefficient, is the best in absolute value. This solution was then retained in the best case.

2. *No Cache/No Proxy Case (Worst Case)*

As in the previous case, the $T_{id}=f(T_{ai})$ and $Time=g(T_{ai}+T_{id})$ correlations were selected. For the first, using linear approximation, the correlation was obtained by the following equation:

$$\boldsymbol{T_{id} = 0.0429* T_{ai} - 185.65} \tag{5}$$

For the second relation which binds *Time* and $T_{ai} + T_{id}$, a series of experiments were carried out in order to obtain the following relation (linear approximation):

$$\boldsymbol{Time = 1.0241*(T_{ai} + T_{id}) + 216.49} \tag{6}$$

By combining (5) and (6), the relation that connects *Time* to T_a must be:

$$\boldsymbol{Time = 1. 068* T_{ai} + 26.36} \tag{7}$$

This relation was implemented so that *SA* can dynamically estimate the *Timeout.*

Finally, to be able to compare these results to the reference protocol, it was necessary to see the variations compared to the effective execution time and, using some weighting, the adequate approximation of *Timeout* was obtained and adopted for the chosen protocol. The coefficients for the basic protocol will be those previously indicated. The comparison of the mean deviation of the estimates of *Timeout* from the real execution time of both protocols showed that the basic protocol gives good performance using a weighting of 2, however, it still penalizes a slower platform. in the other hand, our estimate does not penalize a less powerful platform nor does it give a fast platform an advantage and it only uses a weighting of 1.001. Thus, coefficients 1.001 and 2 were respectively adopted for the protocol and the basic protocol.

4.3 Evaluating the Implementation

The performance of this protocol was evaluated in relation to the basic protocol. The following aspects were taken into consideration:

***Quality of the Approach(*agent re-execution*)*:** After comparing this protocol and the basic protocol [1], another comparison was performed in a hostile environment (a platform which intentionally re-executes the code). In the worst case the correction coefficients remain the same: 1.001 for this protocol and 2 for the basic protocol. Then, the detection rate was measured. The two protocols show a 100% detection rate. The rates are certainly satisfactory. However, the variation of the *Timeout* is very large in the case of the basic protocol because the re-execution of the code for the mobile agent requires simply the re-execution of its sensitive code (*critical()*) which is in fact a small function compared to the total execution time of the *MA*. In other circumstances, if the *MA* is carried out on a slow platform, the protocol will probably be erroneous considering that the *Timeout* estimate is too short and will thus penalize less powerful platforms. In the second case, the rectification coefficients of 0.975 for this protocol and 0.65 for the basic protocol were adopted. A percentage of 35.3% was obtained as a rate of detection for the basic protocol and 100% for this protocol, which confirms both the refinement and the quality of this approach. Moreover, this protocol has a reasonable standard deviation compared to the execution time of the malicious platform. Thus, it could not be erroneous in terms of the integrity of a slow platform. At the same time, it will avoid a very powerful and malicious platform in order to pass unnoticed.

***Analysis of Traffic Generated*:** The amount of traffic generated between the various participating platforms is usually (43%) due to the exchanges between the *O* and *T* platforms of origin. This is due to the fact that on platform *T,* two entities are carried out: the agency which carries the *SA* agent and the *Region* to which all the agencies, places and agents of the protocol belong. Thus, all the migrations will have to be deferred there (to the register of the region). Considering that migrations are weak (by instantiation and not an entire migration of the code), the agents are thus instantiated on all the platforms that have been visited, whereas the code remains on the platform of origin. This platform will have to send packages to *T* to update the register of the *Region.* In addition, the traffic generated due to the exchanges between *T* and P_1 is equivalent to the traffic between *T* and P_2. This is due to the fact that both P_1 and P_2 perform the same role. However, the protocol adds 13% to the amount of traffic due to the *SAR*, but this traffic could be partially compensated by eliminating agent estimator *AE* of the basic protocol. Note that this protocol presents an 11.7% increase of the size of the *SA* compared to the basic protocol, while the *SAR's* size (14 KB) is double the size of the estimator agent. In part, that constitutes the price to pay to have a robust protocol.

5 Conclusion

This paper presented a secure protocol for mobile agents. This protocol combines the reference behavior, cooperation between agents, cryptography and timed execution techniques. The reactivity of the protocol makes it possible to mitigate and rectify the damage caused by permanent code modification. Since a possible breakdown of the trusted platform will certainly stop the cooperation, a second agent was introduced to watch for the breakdown of the first cooperating agent. This will guarantee a continuity of the protocol since the agent will continue its mission under full security.

Future research shall address the limitations of this protocol. An in-depth treatment of the timer for denial of service detection will make it possible to obtain a complete measure of detection and protection against this type of attack, which partly facilitates the estimation of the *TimeoutSA*. In addition, a replication was implemented in only one direction between the *SA* and the *SAR*. However, it would be better *SA* and *SAR* get identical roles. Thus, in the case of restoration of the *SAR* after a breakdown of *T*, the protocol will have to update the *SA* after its recovery with all the events which occurred in its absence to perform a handoff.

References

1. A.El Rhazi, S. Pierre, H. Boucheneb, "Secure Protocol in Mobile Agent Environments", to appear, *International Journal on Mobile Information Systems.*
2. G. Allée, S. Pierre, R. H. Glitho, "An Improved Itinerary Recording Protocol for Securing Distributed Architectures Based on Mobile Agents", to appear, *International Journal on Mobile Information Systems.*
3. T. Sander and C.F. Tschudin, "Protecting Mobile Agents Against Malicious Hosts", *in Mobile Agents and Security,* G. Vigna (Ed.), LNCS 1419, Springer-Verlag, 1998, pp. 44-60.
4. 4. G. Vigna, "Cryptographic Traces for Mobile Agents", *In Mobile agents and security,* G. Vigna (Ed.), LNCS 1419, Springer-Verlag, 1998, pp. 137-153.
5. W.M. Farmer, JD Guttman, and V Swarup, "Security for Mobile Agents: Issues and Requirements", in *Proceedings of the 19th National Information Systems Security Conference*, October 1996, pp. 591-597.
6. F. Hohl, "Time Limited Blackbox Security: Protection of Mobile Agent from Malicious Hosts", in *Mobile Agents and security*, LNCS 1419, Springer-Verlag, 1998, pp. 92-113.
7. F. Hohl, "A Framework to Protect Mobile Agents by Using Reference States", in *Proceedings 20th International Conference on Distributed Computing Systems*, IEEE Computer Society, Los Alamitos (Ed.), California, pp. 410-417, 2000.
8. Institute for Applied Information, "The IAIK Java Cryptography Extension (IAIK-JCE) 3.03", in *Processing and Communication of the Technical University Graz*, 2002. Online documentation: http://jce.iaik.tugraz.at/products/01_jce/documentation/javadoc/
9. IKV++, "Grasshopper, Release 2.2, Basics and Concepts (Revision 1.0)", March 2001.
10. http://www.softwareresearch.net/site/teaching/SS2004/MC/BasicsAndConcepts2.2.pdf
11. W. A. Jansen, T. Karygiannis, "Mobile Agent Security'', in *NIST Special Publication 800-19*, National Institute of Standards and Technology, August 1999.
12. W. A. Jansen, "Countermeasures for mobile agent security", in *Computer Communications, Special Issue on Advanced Security Techniques for Network Protection*, Elsevier Science BV, Nov. 2000.
13. N.M. Karnik, A.R. Tripathi, "A Security Architecture for Mobile Agents in Ajanta", in *Proceedings 20th International Conference on Distributed Computing Systems*, Los Alamitos Ed., IEEE Computer Society, California, 2000, pp. 402-409.

14. E. Bierman, E. Cloete, "Classification of Malicious Host Threats in Mobile Agent Computing", in *Proceeding of the 2002 annual research conference of the South Africa Institute of Computers Scientists and Information Technologists on Enablement through Technology* (SAICSIT'02), September 16-18, 2002, pp. 141-148.
15. V. Roth, "Mutual Protection of Co-Operating Agents", in *Secure Internet Programming*, Vitek and Jensen (Eds.), Springer-Verlag, 1998, pp. 26-37.
16. Y. Minsky, R. Van Renesse, F. Scheinder, S. Stoller, "Cryptographic Support for Fault-Tolerant Distributed Computing", in *Proceeding of the 17th ACM SIGOPS*, European Workshop, 1996, pp. 109-114.

A Cross-Layer Approach for Publish/Subscribe in Mobile Ad Hoc Networks

Marco Avvenuti[1], Alessio Vecchio[1], and Giovanni Turi[2]

[1] Dip. di Ingegneria dell'Informazione, Università di Pisa,
Via Diotisalvi 2, I-56122 Pisa, Italy
marco.avvenuti@iet.unipi.it, a.vecchio@ing.unipi.it
[2] IIT Institute, National Research Council (CNR),
Via G. Moruzzi 1, I-56124 Pisa, Italy
giovanni.turi@iit.cnr.it

Abstract. In the context of ubiquitous and pervasive computing, publish/subscribe middleware is gaining momentum due to its loosely coupled communication scheme. In this paper we present Q, a publish/subscribe service conceived to operate over mobile ad hoc networks. With Q, the overlay network that routes events from publishers to subscribers dynamically adapts to topology changes by means of cross-layer interaction. Q also supports content-based filtering of events through mobile code: subscribers can specify in detail the notifications they wish to receive by defining proper filter classes, then binary code of filters is exchanged during runtime by participating nodes.

1 Introduction and Background

Publish/subscribe middleware supports the construction of event-based systems, whereby generators publish event notifications to the infrastructure and consumers subscribe to receive relevant notifications [1]. The interaction model between communicating partners is asynchronous by nature, and is therefore particularly attractive in the context of highly dynamic systems such as mobile ad hoc networks (MANETs). The simplest way to build such infrastructure is through a centralized server, in charge of collecting events and dispatching notifications to interested parties. Clearly this solution, besides its limited scalability, is impractical within MANETs, as these networks are intrinsically peer-to-peer. This suggests to distribute the notification service over the participants, enabling each entity to manage subscriptions and forward events. Unfortunately, most distributed event notification systems assume a fixed network infrastructure and are not aware of the underlying network topology, and hence are likely to cause unnecessary traffic when used in MANET environments.

1.1 Related Work

Here we summarize some relevant proposals of publish/subscribe middleware for MANETs (for the sake of brevity, we omit those systems that have not been specifically designed for infrastructureless networks).

STEAM [2] is a middleware service based on the idea that communicating entities are likely to interact once they are in close proximity. In other words, events are valid within a certain geographical area surrounding the producer: this limits forwarding of event messages producing positive effects on the usage of communication and computation resources. The system supports three different ways of filtering events: type filtering (on the base of the type, i.e. a name, of events), content filtering (to specify the values of event fields), and proximity filtering (to specify the scope within events are disseminated).

In [3], the authors identify and address the design challenges encountered in the implementation of a Java Message Service (JMS) solution for MANETs. To cope with the lack of readily available routing protocols integrated into the network layer, the system includes an application-level implementation of ODMRP [4] (a multicast routing protocol) as message transportation system.

In [5], the authors propose a technique useful to rearrange efficiently the routes traversed by events in response to changes in the topology of the network of dispatchers. The reconfiguration algorithm assumes that subscriptions are forwarded on an unrooted tree topology that connects all dispatchers.

EMMA [6] is an adaptation of JMS for mobile ad hoc environments where the delivery of messages is based on an epidemic routing protocol. When a message is sent, it is replicated on each host in reach, then if a node that contains a copy of the message, during its movement, gets in touch with other nodes, the message spreads to these nodes as well. Proper mechanisms are used to avoid message duplicates.

1.2 Contribution

This paper presents the design of Q[1], an infrastructure for publish/subscribe conceived to operate in MANET environments. Q is a type-based system: events are instances of application-defined types, and both publishers and subscribers have to specify the type of events they produce or are interested in.

We briefly recall some properties of type-based publish/subscribe [7]. First, it preserves type safety and encapsulation through application-defined event types. Second, it easily integrates with object-oriented languages where event types are mapped to classes, and the hierarchical organization of event types is reflected by the inheritance relationship between classes. Third, it does not preclude content-based filtering of events: during subscription an application can narrow the set of events of interest by providing a filter, expressed as a set of conditions that must be satisfied by event fields.

Besides these features, Q presents two other distinguishing characteristics: reconfiguration through cross-layer interaction and content-based filtering by means of mobile code.

Cross-layer interaction: As in classical protocol stack architectures, ad hoc network activities can be organized in layers. In this model, each layer in the protocol stack is designed and operated separately, with static interfaces independent from

[1] Q is the title of a Luther Blisset's novel. Q is also the name of one of the main characters, a spy.

network constraints and applications. However, there are some functions which cannot be assigned to a single layer, but are part of an optimization strategy targeting the whole stack. Cross-layering is an innovative form of protocols interaction, placed beside strict-layer interactions, which makes these optimizations possible [8]. Picking just one example, the routing protocol plays an important role, as it collects network topology information. Sharing topology data, potentially simplifies tasks taking place at other layers (e.g overlay network), avoiding explicit communication to collect it again.

Q interacts cross-layer with routing agents at the network layer, and uses topology information to obtain a self-reconfigurable overlay network that increases communication's efficiency: application-level routes connecting publishers to subscribers closely reflect unicast routes at the network layer.

Content-based filtering through mobile code: Content-based filtering of events can greatly reduce the amount of traffic generated by publish/subscribe systems. This is particularly significant within a MANET scenario where network resources are scarce. In Q, filters are instances of application-defined filter classes. Filter objects, that are issued by event consumers during subscription, are moved towards the publishers of relevant events to maximize the filtering effects. Since filter classes are application-defined, it may happen that a node that receives a filter object is not in possession of the corresponding class. Q is provided with mechanisms that support code mobility, so that filter classes can be downloaded at runtime by participating nodes when needed.

2 Q's Overlay Network

Delivery of event notifications requires the cooperation of nodes located between publishers and subscribers, that are involved in the forwarding activity. In the following we will refer to "dispatcher" nodes as those actively involved in event forwarding, while we will say that a node becomes a dispatcher to mean that it starts forwarding event notifications. For the sake of simplicity, we assume that all nodes that compose the network are able to participate to the protocol.

2.1 Connecting Publishers and Subscribers

In a publish/subscribe system communication is anonymous, i.e. publishers and subscribers have no knowledge of each other identity [9]. A publisher injects an event into the system without specifying the recipients, then the dispatching service is responsible of the delivery of such notification to all interested parties. In the absence of a centralized entity, this poses the problem of establishing routes to forward event notifications from publishers to subscribers. As highlighted in [1], there is the need of broadcasting some information because participants identity is not known. The first solution consists in broadcasting event notifications, which is clearly inefficient since notifications are potentially very frequent, and they would be delivered also to uninterested nodes. The second alternative broadcasts subscriptions, while notifications get routed through shortest paths. The last alternative, adopted by Q, sees publisher nodes broadcasting

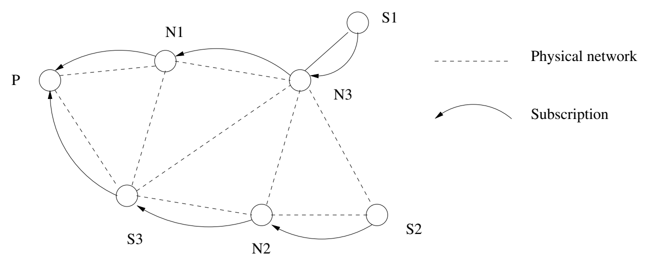


Fig. 1. Network with three subscribers (*S1*, *S2*, and *S3*) and one publisher (*P*). Arrows show the direction of subscriptions (events flow in the opposite way). One-hop communication links are also shown.

event advertisements, and interested nodes replying back with subscriptions. This message exchange sets up the routes that will support event notifications.

In Q, because of the dynamic nature of MANETs, advertisements must be periodically retransmitted, with period T_p, in order to tolerate the loss of messages. Each advertisement contains the publisher's identity (i.e. its IP address and port number) and the type of the events it generates. Upon receiving an advertisement, a node stores in a local table i) the type of the event, ii) the ID of the publisher, iii) the ID of the direct (one hop) neighbor from which the advertise message was received. Afterwards, it re-transmits the message to its neighbors. The first two fields are used to keep track of active publishers in the network and the type of produced events, while the third is used to route subscriptions of interested nodes back to the publishers. Entries are discarded after a time equal to $2T_p$.

If subscriber *S* is interested in events generated by publisher *P*, a chain of dispatchers must be established to forward events from *P* to *S*. The chain is initiated by *S*, which uses a subscribe message to register as an event consumer with the neighbor *N*, from which it received *P*'s advertisement. Afterwards, *N* registers as an event consumer with the direct neighbor from which it received the advertisement and so on until a path is built from *S* to *P*.

As there could be many subscribers interested in events advertised by *P*, the system tries to aggregate dispatcher chains whenever possible: a node already acting as a dispatcher for an event type *E*, does not further forward subscriptions for *E*. In the scenario shown in Fig. 1, the system needs to activate some nodes as dispatchers in order to forward events from *P* to the three subscribers. In particular, nodes *N1* and *N3* are activated to forward events from *P* to *S1*. *S3* directly receives events from *P*, while *N2* and *S3* forward events towards *S2* (i.e. *S3*, besides acting as a sink for events, also operates as a dispatcher).

2.2 Reconfiguration

As nodes mobility induces topological reconfigurations of the physical network, Q has to reconfigure the overlay network to contain the cost of event dispatching. This is done by means of PING messages, which are periodically generated by

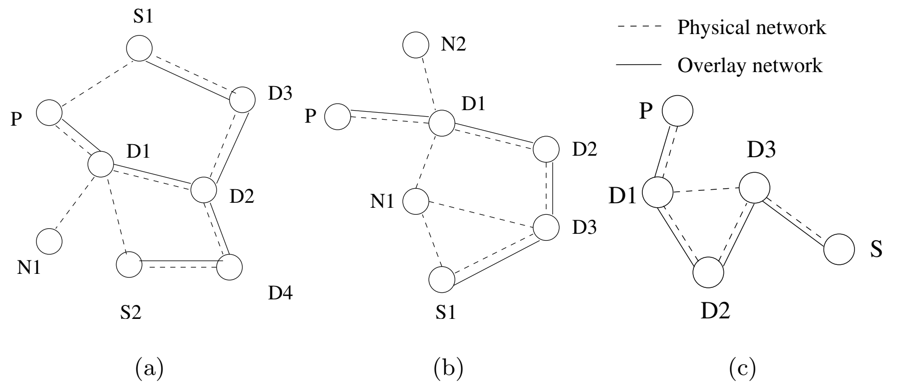


Fig. 2. Examples of reconfiguration. a) *S1* unsubscribes from *D3* and becomes a direct consumer of *P*, while *S2* unsubscribes from *D4* and becomes a consumer of *D1*; b) *S1* unsubscribes from *D3* and becomes a consumer of *N1*, which is activated as dispatcher; c) *D3*, a dispatcher, starts a reconfiguration process by unsubscribing from *D2* and subscribing to *D1*.

each publisher (with period T_{ping}). PING messages are forwarded the same way of regular events, but at each hop the dispatcher appends its own identifier before sending it to its consumers. This makes each dispatcher aware of its level in the forwarding chain (i.e., the publisher is a level 0 dispatcher, which communicates with level 1 dispatchers, and so on). Also, each node becomes aware of the current path from the publisher to itself. This information, together with the content of the routing table, is used to understand if the chain of dispatchers has to be reconfigured or not, as explained in the following.

Cross-Layer Interaction. The current implementation of Q is on top of Dynamic Source Routing (DSR) [10], a source-based routing protocol. This means that the routing table of each node contains the list of nodes that messages have to traverse to reach a given destination.

On receiving the PING message, each node analyzes the list of upstream dispatchers and, for each node in the list, i) calculates the distance at the application-level, i.e. the number of intermediate dispatchers, between itself and that node; ii) retrieves from the routing table the distance between itself and that node in terms of physical hops. Then, the node evaluates if a solution better than the current one exists. In that case it performs a partial reconfiguration of the network by unsubscribing from its current upper-level dispatcher and subscribing with another node in the list. In the following we give an intuitive description of the algorithm by means of some examples.

Let us consider the network in Fig. 2(a): when *S2* receives the PING message generated by *P*, it sees that *D1* is one hop far away at the network level and three hops far away at the level of the overlay network. Therefore, *S2* unsubscribes from *D4* and connects to *D1*. The same way, *S1* discovers that the producer of the events is a direct neighbor. Then, *S1* unsubscribes from *D3* and becomes a direct consumer of *P*. Since *D4*, *D3*, and *D2* are no longer used to route events, they stop serving as dispatchers for that type of events.

The same technique can be applied also when an upstream dispatcher is not directly connected, at the network level, to the subscriber that performs the reconfiguration. In this case, the entry in the routing table indicates how many hops are needed to reach that node and which other nodes are involved. For example, as shown in Fig. 2(b), *S1* can understand that *D1* is two hops far away, and that the distance at the application-level is three hops. Therefore, *S1* can improve the topology of the overlay network by unsubscribing from *D3*, and subscribing to *N1*, which in turn must subscribe to *D1*. This operation is achieved through a special message (*Subscribe&Forward*, SF) which is sent by *S1* to *N1*. The SF message contains the list of all nodes on the route between *S* and *P* that must be activated as dispatchers.

As intermediate dispatchers are as a matter of fact "indirect" subscribers, they run the same reconfiguration procedures. For example, as shown in Fig. 2(c), on receiving the PING, *D3* decides to unsubscribe from *D2* and connects to *D1*, a direct neighbor.

Communication to a Dispatcher Becomes Multi-Hop. It may happen that two nodes that are adjacent in the overlay network become able to communicate with each other only through the intervention of another node acting as router. For example, as shown in Figure 3, dispatchers *D1* and *D2*, that are adjacent in the overlay network, become able to communicate only thanks to node *N3* (because of the movement of nodes). Let us suppose that PING messages flow from *D1* to *D2*: when the latter receives the PING, it becomes aware that *D1*, its upstream dispatcher, is reachable by means of node *N3*, which is not active as a dispatcher. Then *D2* sends a SF message to *N3*. *N3* becomes an active dispatcher and subscribes with *D1*. At the same time *D2* unsubscribes from *D1*.

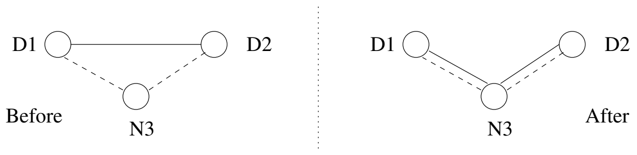


Fig. 3. Communication between D1 and D2 becomes multi-hop: N3 acts as a router. The overlay network is reconfigured by D2 that unsubscribes from D1 and sends a SF message to N3, which becomes a dispatcher.

With this strategy, all nodes that route events at the network layer become dispatchers for that type of events (basically, they become routers for that type of events also at the application level). This facilitates the reuse of forwarding chains, as that nodes become possible attach points for other branches.

2.3 Node Exit and Failure

Because of the error-prone nature of MANETs, the system must be able to tolerate not only the voluntary exit of a node from the overlay network, but also involuntary disconnections. Voluntary exit of publishers and subscribers is supported by two explicit messages, *stop publishing* and *unsubscribe* respectively. Involuntary disconnections are detected by means of PING messages.

3 Content-Based Filtering Through Mobile Code

A prototype of Q has been implemented for the Java 2 SE platform. Besides the mechanisms used to build and maintain the overlay network, the prototype allows for content-based filtering of events: each subscriber can specify a content filter expressed as a set of conditions on the fields of the type of the event. All events that do not satisfy the conditions specified in the subscription pattern are not delivered to the subscriber.

In many content-based publish/subscribe systems, conditions are expressed by means of specific languages. As pointed out in [11], this approach has some drawbacks: the language used to express subscriptions is different from the one used to program the application and syntax errors violating the grammar of the subscription language are detected only at runtime, when conditions are parsed.

With Q, filters are expressed by means of the Java language. More in detail, the programmer can define a new application-specific filter as a class provided with the method *boolean matches(EventType e)* that returns *true* when the event *e* must be delivered to the subscriber, *false* if *e* has to be discarded. The body of the *matches()* method can make use of all public fields and methods of the event object passed as argument.

For example, if a subscriber is interested in receiving events related to the temperature (instances of the *TemperatureEvent* class), but only if the value is greater than a given threshold, the *TemperatureFilter* class could have the following structure (*Filter* is the base class of all filters):

```
public class TemperatureFilter extends Filter {
    private int temp;
    public TemperatureFilter(int t) { temp = t; }
    public boolean matches(EventType e) {
        if (!(e instanceof TemperatureEvent)) return false;
        TemperatureEvent tev = (TemperatureEvent)e;
        if (tev.temp > temp) return true;
        else return false;
    }
}
```

Then, to receive temperature related events when the value is greater than 30 degrees, the subscription operation can be performed as follows:

```
TemperatureFilter tf = new TemperatureFilter(30);
subscribe(tf, ...);
```

3.1 Composition of Filters

In many situations, there is no need to build application-specific filters from scratch. More easily, they can be derived from the composition of elementary filters (for example from a library). Q supports the composition of filters by means of specific classes. For example, the *AggregateOrFilter* class allows the programmer to define a new filter as the OR function of an arbitrary number of elementary filters *f1*, *f2*, ..., *fn*. In other words, the *matches()* method of the *AggregateOrFilter* class returns *true* if at least one of the filters *f1*, *f2*, ..., *fn* returns *true*, otherwise it returns *false*. The skeleton of the *AggregateOrFilter* class can be defined as follows:

```
public class AggregateOrFilter extends Filter {
    private ArrayList filterList;
    public AggregateOrFilter() { filterList = new ArrayList(); }
    public boolean matches(EventType e) {
        for (int i=0; i<filterList.size(); i++)
            if (((Filter)filterList.get(i)).matches(e)) return true;
        return false;
    }
    public boolean addFilter(Filter instance){
        return filterList.add(instance);
    }
}
```

The *addFilter()* method is used to add elementary filters. Note that the *AggregateOrFilter* class extends *Filter*, which means that instances of *AggregateOrFilter* can be used in turn as elementary filters.

3.2 Mobility of Filters

In a distributed implementation of the event notification service, filters can be used to selectively propagate notifications only through those links that are part of a delivery path from publishers to interested subscribers. To ensure maximum traffic reduction, filtering must be performed as close as possible to publishers, so that event notifications that do not match the subscription of any client can be blocked immediately.

Q pushes filter objects as close as possible to publishers. For example, let us suppose that a subscriber *S* is interested in receiving events of type *K* that match a filter *f*. Let us also suppose that publisher *P* produces events of type *K* and that communication between *P* and *S* is guaranteed by a third node, say *D*. When *S* starts the subscription process, the filter object *f* is transferred from *S* to *D*. Then, *D* sends a copy of the filter object *f* to *P*. From now on, all events produced by *P* that match the filter *f* are forwarded to *D*, which in turn forwards them to *S*. Events that do not match filter *f* are dropped by *P* itself, without generating unnecessary traffic.

When a dispatcher serves multiple subscribers the system makes use of aggregate filters. A dispatcher *D* that manages subscribers *S1*, *S2*, .., *Sn*, that are interested in events of type *K* matched by *f1*, *f2*, ..., *fn* respectively, must receive all

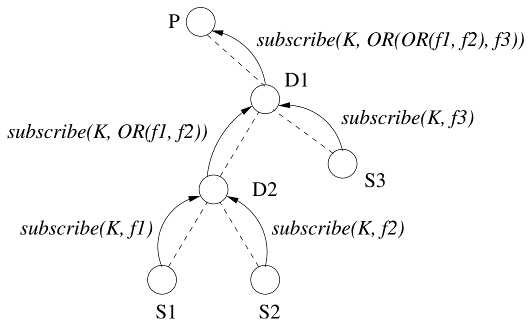


Fig. 4. Propagation and composition of filters

the event notifications that match at least one of the filters of its subscribers. This is done by creating a filter that is the composition (OR) of the filters *f1*, *f2*, ..., *fn*. Then *D* forwards the event notifications it receives only to the relevant nodes. An example is shown in Fig. 4: *D2* propagates a filter that is equal to the OR of *f1* and *f2*, while *D1* propagates a filter that matches the interests of *D2* and *S3*.

Since new filter classes can be defined by programmers, when a node receives a filter object *f*, instance of class *F*, it may happen that the definition of class *F* is not available in the local context. Q supports the transfer of class definitions from the node that issues the subscription to the node that receives the subscription, where definitions are cached.

4 Simulation

The functionalities of the event notification infrastructure previously described have been validated by implementing the system within the QualNet [12] simulator. Communication between members of the overlay network is based on UDP datagrams. Through simulation we also explored the behavior of the system when the following architectural choices are adopted:

Event broadcast: Each time a dispatcher *D* has to deliver the same event message to n consumers it has to retransmit the same data n times. An alternative solution: *D* inserts in the payload of the message the list of consumer addresses, then sends the message only once to the broadcast address, reaching all nodes within its radio range. Every node that receives the message checks if its own address is in the list or not, and decides if the message has to be processed or discarded.

Passive ack for events: If a dispatcher *D1* sends an event message to another dispatcher, *D2*, the latter has to reply with an ack to confirm that the message has been correctly received. Passive acknowledgement can be an alternative solution: if *D2* is not the last dispatcher in the path, it has to retransmit the message to its downstream dispatchers; therefore *D1*, by listening to messages transmitted by *D2*, can understand that an event message has been correctly received by *D2*.

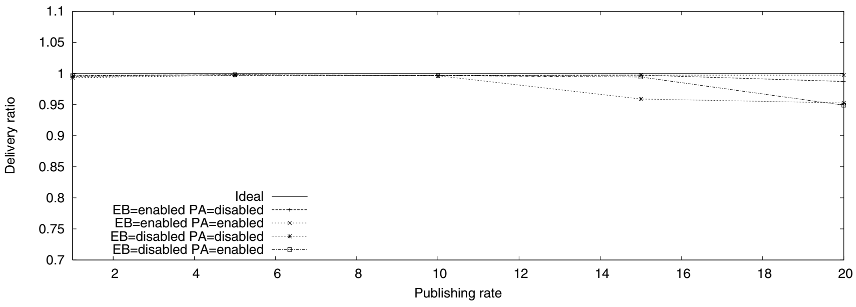


Fig. 5. Delivery ratio of event notifications with 1 publisher and 1 subscriber

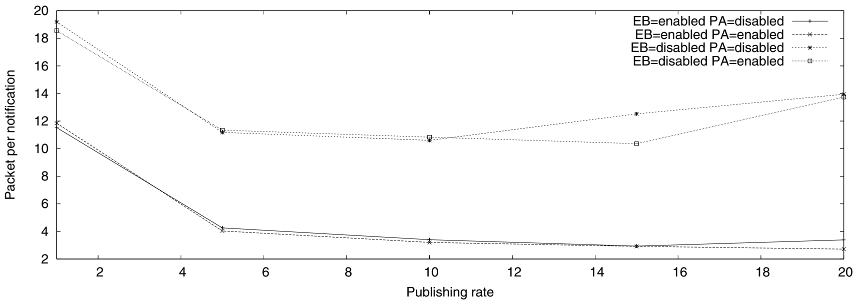


Fig. 6. Packets per notification with 1 publisher and 1 subscriber

Simulations have been carried out according to a scenario where 50 nodes, equipped with 802.11 radio interfaces (2Mbit per sec.), move in a $1000mX1000m$ area. The model for mobility is random waypoint with pause time equal to $30s$ and max speed equal to $10m/s$. Size of event notifications is $512Bytes$. The capability of filtering events on the base of their content is not considered. The performance indexes are the ratio of successfully delivered notifications and the number of packets generated per delivered notification (the latter provides a measure of the overhead of the protocol). Both indexes are evaluated as functions of the rate of publications (events/sec.).

Figures 5 and 6 show the performance of the system with one publisher and one subscriber, while Figures 7 and 8 are related to a scenario with one publisher and five subscribers. The four curves correspond to the alternative solutions mentioned above: the labels indicate if events are sent as broadcast (EB) or not, and if passive acknowledgement (PA) is enabled or not.

In the considered scenarios, the solution that provides the best performance is the one that adopts both event broadcast and passive acknowledgement. This is more evident in the scenario with five subscribers, especially at high publishing rates. In particular, when events are sent as broadcast the number of packets per delivered notification is greatly reduced. Instead, the effects of the passive acknowledgement technique, even if positive, are less evident. Probably this is

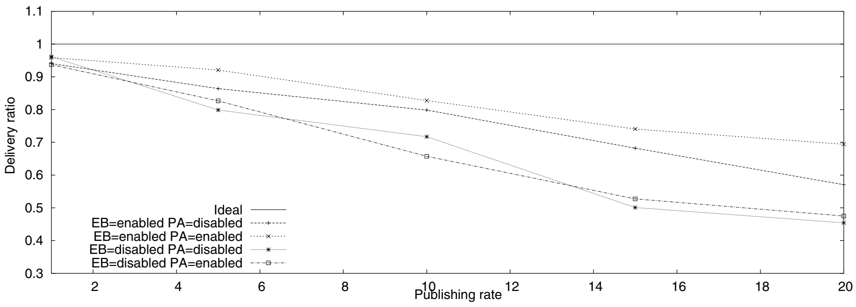


Fig. 7. Delivery ratio of event notifications with 1 publisher and 5 subscribers

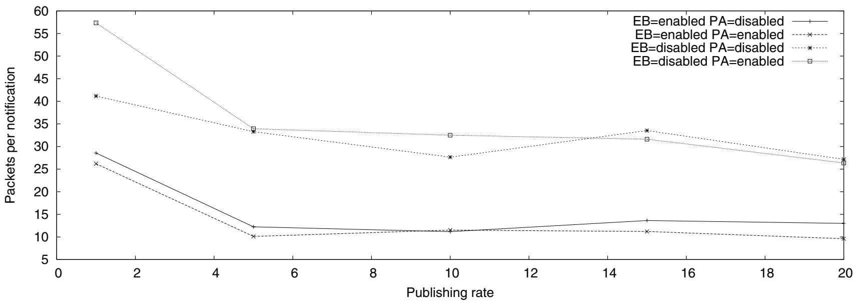


Fig. 8. Packets per notification with 1 publisher and 5 subscribers

due to the fact that, in the considered scenarios, dispatching chains are rarely longer than 2-3 hops (the radio range is approximately $300m$).

5 Conclusion

Due to its decoupled communication scheme, publish/subscribe middleware is particularly attractive in the domain of mobile computing, where the quality of communication and connectivity change dramatically in a short time scale. Q is an infrastructure for publish/subscribe specifically designed for mobile ad hoc environments: the overlay network used to route event notifications adapts to dynamic topology changes by means of information extracted from the routing layer. Currently Q relies on the abstract representation of the network provided by a reactive and source-based protocol. Nevertheless, we believe that with minor modifications it could operate also on top of proactive routing protocols, that provide a richer view of the network (information is not limited to used routes).

Besides its reconfiguration capabilities, Q also supports content-based filtering of events. To achieve seamless integration of filtering with object-oriented languages, in Q filters are instances of application-specific classes. At runtime, filter objects and code are exchanged by participating nodes, maximizing both the benefits of filtering and the flexibility of the system.

The preliminary simulative study demonstrates the feasibility of an event notification service based on a cross-layer approach. The results obtained also provide insightful suggestions about the efficiency of different implementation techniques such as event broadcasting and passive acknowledgement. Future work will concern the evaluation of the performance of Q in comparison with other publish/subscribe systems.

Acknowledgements. This work was carried out under the financial support of the Italian Ministry for Education and Scientific Research (MIUR) in the framework of the Project FIRB-VICOM.

References

1. Carzaniga, A., Rosenblum, D., Wolf, A.: Design and evaluation of a wide-area event notification service. ACM Transactions on Computer Systems **19**(3) (2001) 332–383
2. Meier, R., Cahill, V.: STEAM: Event-Based Middleware for Wireless Ad Hoc Network. In: Proceedings of the 22nd International Conference on Distributed Computing Systems, Washington, DC, USA, IEEE Computer Society (2002)
3. Vollset, E., Ingham, D.B., Ezhilchelvan, P.D.: Jms on mobile ad hoc networks. Volume 2775 of Lecture Notes in Computer Science., Springer (2003) 40–52
4. M. Gerla, G. Pei, S.J.L., Chiang, C.C.: On-demand multicast routing protocol (odmrp) for ad hoc networks. (2000) Internet draft, http://www.ietf.org/proceedings/00jul/ID/manet-odmrp-02.txt.
5. Picco, G.P., Cugola, G., Murphy, A.L.: Efficient content-based event dispatching in the presence of topological reconfiguration. In: Proceedings of the 23rd International Conference on Distributed Computing Systems, Washington, DC, USA, IEEE Computer Society (2003) 234
6. Musolesi, M., Mascolo, C., Hailes, S.: Adapting asynchronous messaging middleware to ad hoc networking. In: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing, New York, NY, USA, ACM Press (2004) 121–126
7. Eugster, P.T., Guerraoui, R.: Distributed programming with typed events. IEEE Software **21**(2) (2004) 56–64
8. Conti, M., Crowcroft, J., Maselli, G., Turi, G.: A Modular Cross-Layer Architecture for Ad Hoc Networks. In Press, C., ed.: Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks. (2005)
9. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Computing Surveys **35**(2) (2003) 114–131
10. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. Mobile Computing (1996) 153–181
11. Eugster, P.T., Guerraoui, R.: Content-based publish/subscribe with structural reflection. In: Proceedings of the 6th Usenix Conference on Object-Oriented Technologies and Systems (COOTS'01), The Usenix Association (2001)
12. Networks, S.: The QualNet Simulator (2003) http://www.scalable-networks.com.

Towards Ambient Networks Management

Marcus Brunner[1], Alex Galis[2], Lawrence Cheng[2], Jorge Andrés Colás[3], Bengt Ahlgren[4], Anders Gunnar[4], Henrik Abrahamsson[4], Robert Szabo[5], Simon Csaba[5], Johan Nielsen[6], Simon Schuetz[1], Alberto Gonzalez Prieto[7], Rolf Stadler[7], and Gergely Molnar[8]

[1] NEC Europe Ltd., Network Laboratories., Kurfürstenanlage 36, 69115 Heidelberg, Germany;
{brunner, simon.schuetz}@netlab.nec.de
[2] University College London, Department of Electronic and Electrical Engineering, Torrington Place, London WC1E 7JE, United Kingdom,
{a.galis, l.cheng}@ee.ucl.ac.uk
[3] Telefónica Investigación y Desarrollo, Emilio Vargas, 6, 28043 Madrid, Spain,
jorgeac@tid.es
[4] SICS, Box 1263, SE-164 29 Kista, Sweden,
{bengta, aeg, henrik}@sics.se
[5] Budapest University of Technology and Economics, Dept. of Telecomm. and Media Informatics, H-1117, Budapest, Magyar Tudosok krt. 2.
{robert.szabo@tmit.bme.hu,simon@david.tmit.bme.hu}
[6] Ericsson Research, Ericsson AB, SE-164 80 Stockholm, Sweden;
johan.nielsen@ericsson.com
[7] KTH – Royal Institute of Technology, Lab. of Comm. Networks;
{gonzalez, rolf}@imit.kth.se
[8] Ericsson Ltd., H-1037 Budapest, Laborc u. 1., Hungary;
gergely.molnar@ericsson.com

Abstract. Ambient Networks (AN) are under development and they are based on novel networking concepts and systems that will enable a wide range of user and business communication scenarios beyond today's fixed, 3rd generation mobile and IP standards. Central to this project is the concept of Ambient Control Space (ACS) and the Domain Manager control function, which manages the underlying data transfer capabilities and presents a set of interfaces towards the supported services and applications. Network Management Systems of Ambient Networks must work in an environment where heterogeneous networks compose and cooperate, on demand and transparently, without the need for manual (pre or re)-configuration or offline negotiations between network operators. To achieve these goals, ambient network management systems must become dynamic, distributed, self-managing and responsive to the network and its ambience. This paper describes the different management research challenges and four complementary solution approaches (i.e. *Pattern-based Management, Peer-to-Peer Management, (Un)PnP Management, Traffic Engineering Management Application Approaches)* that enable efficient management of ambient networks, and the relationships between them, and presents the main results achieved so far.

1 Introduction

New services and access connectivity have been offered by new telecom operators in recent years due to the liberalisation of telecommunications market. However, the availability of such new services and access connectivity to end users is still restricted by the operators with whom the end users have signed an agreement. Thus, end users have not been able to enjoy the full advantage of new services.

The aim of the EU-IST Ambient Network (AN) Project [1] is to develop novel networking concepts and systems that will enable a wide range of user and business communication scenarios beyond today's fixed, 3rd generation mobile and IP standards. This can be achieved by on-demand establishment of inter-network agreements between network operators, without the need for manual pre-configuration and static off-line negotiation [2]. The key concept of inter-network agreement establishment is *network composition*. In ANs, network composition refers to the gain in access connectivity when one or more networks are composed, resulting in an instant availability of any new services to end users at any time anywhere i.e. *ubiquitous connectivity*. This availability is no longer dependent on the particular operator with whom the end users have signed an agreement.

This paper focuses on the requirements and approaches to the management aspects of ANs. AN management systems must work in an environment where heterogeneous ANs compose and co-operate in an on-demand and transparent fashion, without the need for manual pre-configuration or off-line negotiations. Therefore, AN management systems must be dynamic, distributed, self-managing, and responsive to the network and its ambience [3], [22]. This paper is organised as follows: firstly, a summary of the main concepts and characteristics of AN is presented; secondly, the management requirements and challenges of AN are discussed; lastly, the four approaches towards AN management challenges are presented.

2 Ambient Networks

2.1 The Characteristics of Ambient Networks

The term "Ambient Networks" is defined in this paper as a connectivity network with an Ambient Control Space (ACS) presented in the next section. Fig. 1 shows the two important interfaces i.e. Ambient Service Interface (ASI) and Ambient Network Interface (ANI). The ASI is an interface through which services are accessible within ANs. The ANI is an interface through which networks may co-operate and communicate during network composition and de-composition. The major characteristics of AN [3] are as follows:

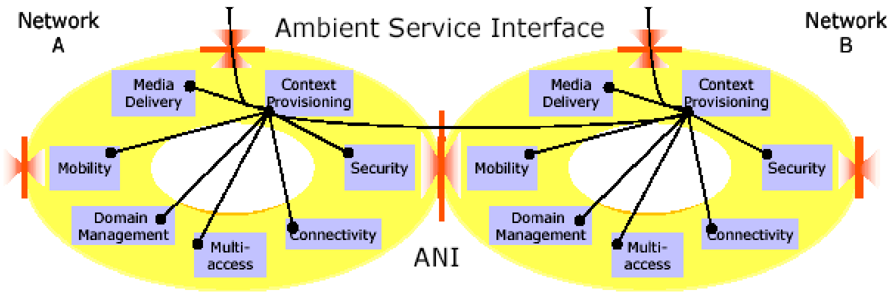


Fig. 1. Ambient Control Space (ACS)

- *All-IP Network*: Ambient Networks are IP-based mobile networks that can be regarded as a continuation of current Internet design practices.
- *Heterogeneity*: Ambient Networks are networks managed and administrated by different operators, and the underlying network technologies may be different.
- *Mobility*: Mobility refers to the network architectures' capability of dynamic composition, as well as the mobility of user groups within ANs. An Ambient Network mobility solution will have to work well across business and administrative boundaries, which requires solutions that ensure the security of inter-domain communications.
- *Composability*: An Ambient Network is capable of dynamically composing (and de-composing) with one or several other network(s). The participating Ambient Networks may belong to different administrative/ economic entities. Thus co-operation between composing Ambient Networks is essential to provide network services in a co-operative as well as competitive fashion. ANI provides the necessary facilities for communication/cooperation across different composing/decomposing ANs.
- *Control Interface Provisioning*: Well-defined control interfaces to other Ambient Networks and to service platforms or applications are provisioned in AN.
- *Explicit Control Space*: This is a provisioning of the main Ambient Control Space functions. Control functions of individual ANs must also be composed during AN composition. However, individual functions must control the same resources as before composition. Resources should not become common assets after composition, but should be treated as an asset that can be traded.
- *Service Accessibility*: New services that are originally not available to end users in a different domain are now accessible to end users at any time/where via the ASI.
- *Autonomic and Self-management*: autonomic system management in the ambient network environments instead of manual pre-configuration or off-line negotiations.

During AN network composition, a common control space must be agreed on by the networks' management systems. This common control space must be consistent throughout the newly created composed network. When the composed network decomposes, the common control space must also be decomposed. This common control space, called ACS, manages the underlying data transfer capabilities and offers a set of interfaces towards the supported services and applications within and across the composed AN. The ACS comprises a complex set of interdependent control functions such as the basic functions for management, security and connectivity, as well as control functions for supporting mobility, connectivity, multi-radio access, smart media routing. These control functions are being developed in the AN project. The Domain Manager control function assures consistent and autonomic co-operation between all control functions within the ACS.

3 AN Management Requirements and Challenges

Handling the composition of heterogeneous ANs is the major challenge of AN management. Prior to composition, each (heterogeneous) AN has its own ACS that is controlled by its own Ambient Management System (AMS). The latter is governed by a set of pre-defined policies that are known to the AN. AN AMSs are heterogeneous. Different AMSs may (de)compose, resulting in a common AMS to manage heterogeneous network elements (that were previously managed by their own management system). Network elements may also manage themselves i.e. self-management. Fig. 2 shows the contrasts.

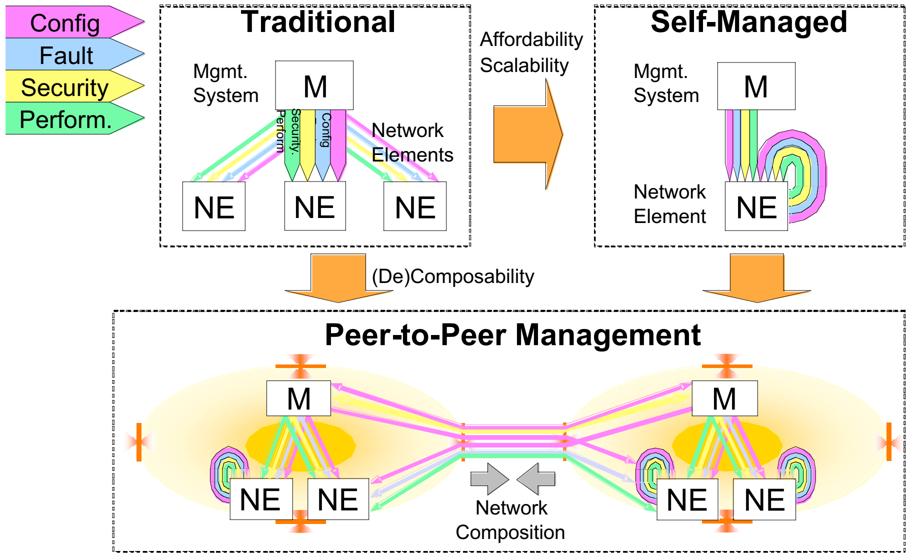


Fig. 2. Traditional management systems Vs. Ambient Management System

During AN network composition, a set of policies control the composition process, and a set of common policies for the new combined ambient control space must be agreed on. Once the networks are composed, the individual ACSs are also composed, resulting in a common ACS. From a management point-of-view, the common ACS is controlled by a common AMS that is governed by the common policies. The term 'composition', therefore, does not just refer to network-layer composition, but also to management-layer composition. This is referred to as *management-layer composition* in this paper.

The AN AMS contrasts with traditional management systems mainly in the area of dynamic heterogeneous management-layer composability and self-manageability. Generally, traditional management systems are homogeneous, and have a clear separation between the roles of management systems and network elements. SNMP or CMIP are generally used, but they do not scale well in large networks, in particularly dynamic (de)composing networks. AN management approaches must be robust, scalable, and adaptable and require less management traffic, less processing load on the management station, shorter execution times, etc. Automatic configuration management is also not supported in traditional approaches. Traditionally, manual configuration has been deployed due to the small scale of network involved. In AN management, however, manual configuration is impractical. Policy-based management was proposed to ease configuration, but the deployment of policy-based management is inhibited by the difficulties in understanding of the concept of policies by practitioners, and the difficulties in standardising configuration management.

Management system composition, through inter-domain interfaces, has already been demonstrated in the TMN model [4][5][6][7]. However, management-layer composition in AN is much more dynamic, and so it requires a simple, efficient, and scalable mechanism for pooling and sharing management information within and across heterogeneous composed networks. The composition mechanism itself should be autonomous and self-managed. This is needed because large or small scale ANs may leave or join on-the-fly i.e. arbitrarily composition and decomposition. The common AMS should be service oriented, that is capable of supporting new services on-demand. The common AMS should consist of distributed AMSs that are co-ordinated through ANIs. The AMSs should be capable of accessing context policies within the AN(s), so that the common AMS may adapt itself to its context based on network conditions or administrative policies. The AMSs should co-ordinate policy negotiation between ANs during composition processes. This also implies that the AMS should resolve conflicts in ANs based on the composition agreement.

From a business perspective, network composition is regarded as a temporary agreement between independent networks (operators) to achieve a common business goal. Thus the AMS should support *automated* creation and administration of network composition. Also, services that were only available in individual AN may now be provided to end users anywhere throughout the composed AN, so increasing business opportunities. The challenge of providing services across continuously composing and decomposing ANs is that autonomic re-configuration of services is needed. This paper suggests that a highly flexible information infrastructure is needed to support automated AN management. This infrastructure should provide on-demand quality-guaranteed connectivity services between AMSs, as well as support for automated management. Although today's Virtual Private Networks (VPNs) provide some QoS guarantees, they suffer from a low flexibility to adapt to the rapid changing i.e. composing and decomposing ANs. Thus, an infrastructure that is secure, dynamic, QoS guaranteed, and programmable for rapid service deployment is required to support the common AMS.

This paper suggests that the dynamic management-layer composition challenge and the self-management challenge of AN may be dealt with through four complementary approaches presented in the next section, that develop and adapt different technologies for the purpose of dealing with these challenges. Initially they are being developed separately within the project but will eventually be integrated through the concept of AMS.

4 Management Approaches and Results

In this section, four approaches towards AN management challenges are presented. The four complementary approaches are: Peer-to-Peer (P2P), pattern-based, Plug-and-Play (PnP), and Traffic Engineering (TE) (also known as Closed-loop-based Approach). The P2P and pattern-based approaches target the management-layer composition challenges, whereas the PnP and TE approaches target the self-management challenges. Due to space limitation, only a summary of each approach and its impact is presented in this paper. For further detail, the reader is directed to the references, in particular [3].

4.1 Peer-to-Peer (P2P) Approach

An AN can be modelled and realized as a Peer-to-Peer (P2P) network [20]. Both P2P networks and ANs are dynamic, decentralized, heterogeneous, and large scale. Thus a management approach that suits P2P networks may be suitable for AN management. Two distinct P2P management applications are presented in this paper. The first application is responsible for the topological composition between management systems of ANs, which organizes peers into the network at different logical levels. The second application is for the dynamic establishment of a secure, QoS-assured, context-aware, self-adapted management service overlay network across composing ANs to ease the process of management-layer composition and new management service deployment.

Hierarchical structures are commonly used to create scalable systems e.g. in routing. A hierarchical management overlay structure is introduced in AN, in which P2P paradigms are used for the maintenance of the hierarchy, as well as for solving management and control tasks. This is called the p-AMO (Peer-to-Peer Ambient Management Overlay), in which nodes are organized as *peers, super-peers* and *peer-groups*. A common management domain is formed by a peer group i.e. a set of peer nodes. Peer-groups support distributed network management. Each peer-group elects a representative, which is known as the *super-peer (SP).* A SP *is responsible for* negotiating with other peer-groups outside the domain. Super-peers of ANs themselves may form new peer-groups at a higher hierarchy level. Super-peer selection can be found in [3]. An example hierarchical overlay network (thick lines mark super-peers) is shown in Fig. 3. A, B and C; D, E and F form different peer-groups that are composing. Note that the super-peers i.e. B and E form a higher-level peer group. There is no limit to the number of hierarchy levels.

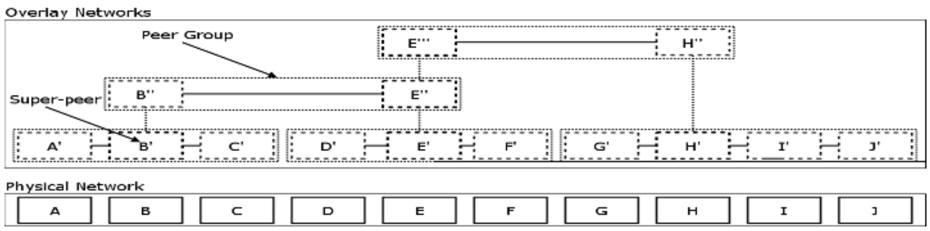


Fig. 3. Hierarchical overlay network representation

Two types of composition model are defined; absorption and gatewaying. Absorption is a full merging of ANs, resulting in one single management domain. This model is only suitable when full trust concerning the management of networks is established between the composing networks. Gatewaying refers to the arrangement in which one or more Gateways is/are selected from the composing ANs. The gateways are responsible for network management communications and with it the control of the exchange of management information between ambient network domains. With this arrangement only relevant information on a per-service, per-management function, or per-user might be exchanged in a controlled way. Refer to [20] for further details of P2P management-layer composition and composition models.

As discussed earlier, a secure, QoS-assured, contextaware, and self-adapted management service overlay network is needed in AN to support management-layer composition as well as new management service deployment. This is known as the *Ambient Virtual Pipe (AVP)*. The arrangement for AVP is shown in Fig. 4.

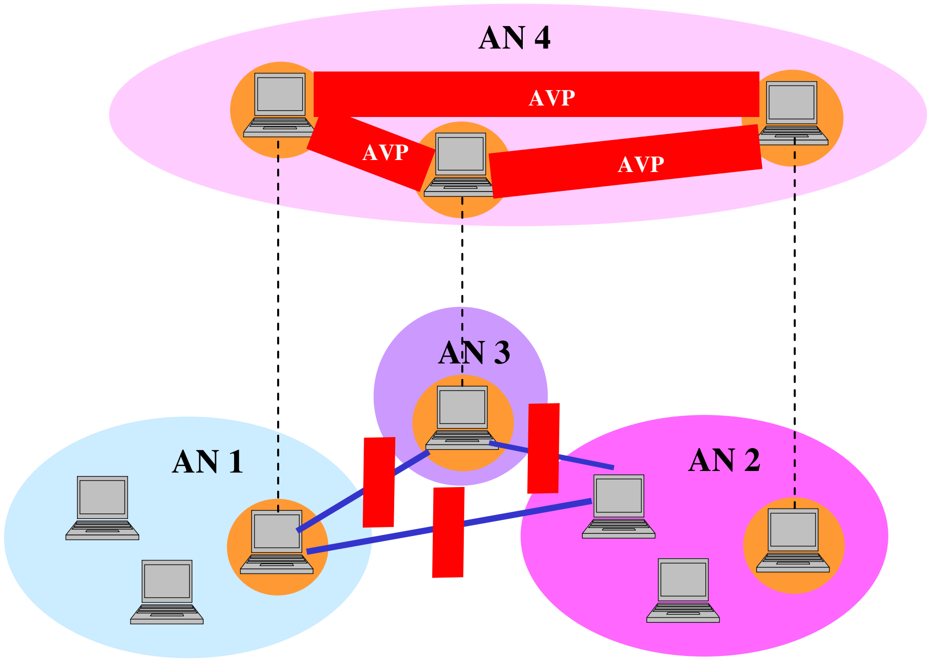


Fig. 4. AVP Establishment and AN Network Composition

The AVP is a *management service overlay network,* established across different underlying ANs. This secure and QoS-assured management service network overlay is essential in AN. Traditionally, a network administrator can only control his own domain, also, management services specific to that domain can only be offered to customers of that domain. With AVP and AN composition, customers of other domains may be offered management services from another domain. Example services are QoS, security and QoC (Quality of Context). AVPs provide the essential QoS and resource assurance as well as security for the protection of management traffic. Note that AVP must be context-aware because of the ever-changing AN topologies, thus AVP requires a flexible and programmable infrastructure [23] for dynamic establishment of AVP as well as self-adaptation to the changing network context. These are achieved by the dynamic injection of active code to desired super-peers for service execution. For instance, QoS is assured through the injection of active code to dynamically prioritise AVP traffic in conjunction with tc and iptable. DINA [21] is used as a programmable platform in AN to support AVP provisioning. Further details can be found in [3].

4.2 Pattern-Based Approach

It was discussed in earlier sections that centralised network management is not suitable for AN management due to the dynamic nature of ANs. The general drawbacks of centralised network management are addressed in [8][9][10]. Dynamic AN network composition requires a common AMS. The composition of individual, heterogeneous management systems require a distributed management approach to gather, control, and coordinate the processing and aggregation of management information inside the individual AN and/or composed AN. This paper suggests that this distributed requirement can be satisfied through a pattern-based approach [11][12] through the use of graph traversal algorithms.

The main benefits of the use of pattern for distributed AN management compared to centralised management are as follows:

- Separation between the semantics of the management or control task from its flow control and topology aware distribution
- Enabling the implementation of scalable management systems for large scale networks
- Facilitating management in dynamic environments such as ANs
- No "prior" knowledge of network topology is required

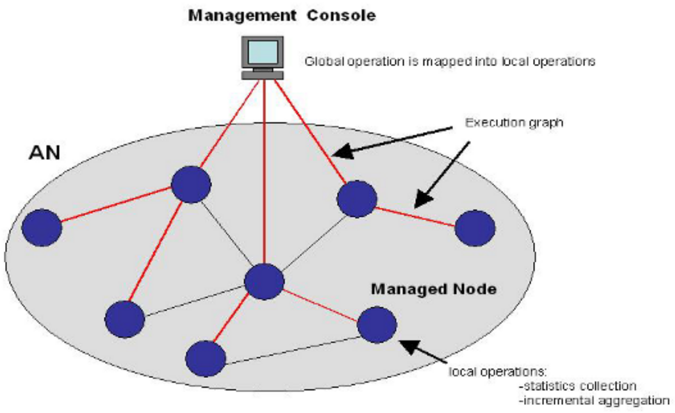


Fig. 5. AN Pattern-based approach

This paper identifies the two application areas where new patterns are applicable. Firstly, patterns can be used to setup and maintain overlay networks in AN; secondly, patterns can be used for real-time monitoring. Two important concepts were developed in the AN project: the *navigation* pattern and the *aggregator*. The aggregator is used for the implementation of the computations that are required to realise the task and are therefore management task specific. On the other hand, the flow of execution of a distributed management operation is controlled by the navigation pattern, thus distributed control is implemented by this generic graph traversal algorithm. The navigation pattern is described by an asynchronous network algorithm. The algorithm can be used for analysis of the pattern's complexity and scalability properties. The navigation part can be reused for various tasks and is a base functionality independent of the task itself.

Network-wide operations can be mapped into local operations through the use of pattern-based applications. Once mapped, local operations can be carried out by managed nodes. An example network-wide operation is the identification of top X traffic flows in terms of bandwidth used. The task is split and distributed using navigation patterns, resulting in an execution graph. For other tasks such as monitoring, local operations are for instance statistics collection and incremental aggregation. The aggregation is asynchronously done in parallel i.e. all nodes contribute to the calculation (see Fig. 5).

Setting up and maintaining overlay networks in dynamic (de)composing ANs are challenging tasks. Potential overlay nodes that are suitable are detected by a path-directed search pattern prior to setting up overlay networks in AN. The overlay network calculation (optimisation) is carried out at the node where the search pattern terminates. The pattern for the setup is a fully-routed pattern. The setup pattern will deliver a setup message to the chosen nodes and the chosen nodes will set themselves up. The AN overlay network has specific properties such as that a path between a source and a destination should include certain nodes with certain functionalities in a certain sequence. Functionalities in the Ambient Network context mainly refer to multimedia processing such as caching, transcoding, adaptation, or intelligent routing.

The searches for potential Overlay Nodes use a path-directed search algorithm, where the concept of dynamic search along the path is used. The basic idea of this algorithm is to limit the scope of the search to a configurable area along the end-to-end path (given traditional IP routing) between communicating peers. It starts from the source node and expands along the end-to-end routing path towards the destination nodes with a sideway expansion of a given distance (e.g. based on the number of hops, delay, or other parameters). Depending on the type of resource, function, or service that is searched, this parameter can be changed. Since the multimedia processing functions are not assumed to be directly on the path, but should ideally be located near the path, the path-directed search pattern is ideal for locating appropriate overlay nodes on demand.

Mechanisms for maintaining overlay network functions are proposed in AN. The source node periodically initiates a search pattern and either selects best nodes, or stores the gathered information and performs the selection when the need arises. It then reconfigures the overlay network either at the end of a specified interval or when it receives a reconfigure request from one of the overlay nodes. This mechanism requires overlay nodes to send keep-alive messages to their neighbours. When a node fails to receive a keep-alive messages from its neighbour within a specified period of time, the node itself initiates the search instead of sending a request to the source node. Because there might be more than one neighbour to the dead node there should be a mechanism for choosing the neighbouring node that will initiate the search, for instance the node with the highest node ID. Also, as an interim solution until the new network is setup, the node can recompute the shortest path, omitting the dead node from the computation.

Note that various other functions could be implemented using the same or similar communication patterns. The existing functionality simply needs to be replaced by new code or policies in order to perform different maintenance steps. Also the description above only examines the removal of nodes, but various other types of degradation can be envisaged. For instance, when due to a degradation of a wireless link underlying a virtual link degrades, the overlay might be changed. Note that adaptation within the overlay is further addressed in the P2P approach and [14]. The AN pattern-based approach further addresses special requirements of the use of pattern in a mobile environment: *overlay maintenance*, *security*, *robustness* and *mobility support*. Details of the AN pattern-based approach can be found in [3].

4.3 Plug-and-Play (PnP) Approach

PnP management is responsible for automatic and complete configuration of AN elements. It enables new AN elements to become an integral part of AN domains. This includes not only the automatic configuration of end-host interfaces as known in current networks but also complete configurations of router elements, base stations, or wireless access points when they are attached to AN domains and initialisation/maintenance of a hierarchical peer structure for management purposes. AN elements may arbitrarily attach to or detach from AN domains, thus incurring dynamics in network topology. Therefore, PnP management not only has to configure newly attaching AN elements, but also to perform constant or periodic verification and optimisation of current configurations. Additionally, PnP management must interact with other approaches. A new AN element may interconnect two or more different AN domains. This has to be reported to the composition component, which possibly triggers new network compositions, using either the gatewaying or the absorption approach. Note that we differentiate between AN elements joining a network in a Plug-and-Play manner, and AN composition, which deals with the composition of whole networks. PnP management will support AN composition only by providing operable network connections in between composing ANs, but it does not perform full composition of the corresponding ACSs. It can be seen as the first step towards composition. The composition task is left to the management-layer composition of the P2P approach.

Similarly, a newly configured AN element modifies the existing network topology. Therefore, PnP management should inform the TE management, because the new element may provide additional options to increase network performance. When P2P Management has to setup a new overlay network on top of already configured networks, it may reuse PnP components to configure the overlay network as well. So running the PnP mechanism within an overlay helps easy configuration. However, in most cases within an overlay network, special routing, naming, and addressing is used, and therefore configuration is already performed at overlay setup time. The PnP management requires a basic set of components, each managing a part of an AN element's configuration:

Base Station/L2 configuration: The Base Station/L2 configuration component is responsible for providing operable link layer configurations that serve as communication channels between AN elements. Without any link layer connection, no communication is possible at all. In the case of a wireless access point, this task involves configuration of a proper ESSID, selection of wireless channel, adjustment of signal power, etc.

IP client/Locator configuration: PnP configures interfaces associated with available link layer connections to provide basic network connectivity. In IP-based networks, an appropriate IP address has to be retrieved for such interfaces. PnP management handles detection of existing AN domains, which may predetermine IP/Locator settings. It is also responsible for duplicate address detection in case of network compositions. The IP client/Locator configuration enables the AN element to act as an end-host. It is then able to contact or be contacted by other elements of its current AN domain. Note that AN locator denotes the location of an interface within a network, and it is distinct from the identifier of the node or interface. In IP networks normally both functions are used together in the IP address.

Router configuration: PnP management configures all available interfaces of the AN element. It has to prepare and manage address spaces for these interfaces. To perform this task an AN element uses basic connectivity to existing AN domains, which is provided by the IP client/Locator configuration. AN elements within these AN domains support the configuration process with information that is already available in the working part of the AN network.

Routing configuration: Once the AN element has configured available interfaces, it starts acting as a router. Therefore, it must configure and activate an appropriate routing protocol, e.g. OSPF. Basic configuration parameters can be retrieved from already-running routers within the AN domains that the AN element connected to in the previous steps.

Super-Peer election: AN P2P management relies on a hierarchical structure to ensure scalability of the system. To avoid single point of failure and increase robustness, the management information is distributed between the peers, using redundant distributed hash tables. The election of the Super Peer from the peers of an AN, the maintenance of the distributed hash table and the replacement of the Super Peer upon the failure of the earlier elected one should be transparent to the user. These tasks rely on the inter-peer communication, which requires valid routing tables in the routers.

Running each of the PnP components for a new AN element in the order stated above will allow the AN element to become an integral part of AN domains. However, this set of components has not only to be executed once per new AN element. Due to the dynamics in ANs, each component must constantly or periodically adapt and optimise the AN elements' current configuration. The Base Station/L2 configuration component might have to reconfigure Base Station parameters, e.g. due to two approaching AN networks using the same wireless channel. In some cases, IP/Locator configuration component has to reassign an interface's IP/Locator, e.g. if detection of duplicate addresses occurs. The router configuration component can change address spaces, e.g. when running out of addresses due to many connecting clients. The routing configuration component has to reconfigure routing protocol parameters when topology changes for any reason. Last but not least, the information base of the Super Peers should be updated as the status of the peers change within the AN and a new Super Peer should be drafted if the earlier elected one fails (or more likely leaves the network).

In general, the configurations of different components are not independent of each other. Consequently, the PnP components have to interact with each other to build a consistent configuration. In particular, the PnP components have to implement the following interactions:

1) A network interface that established a new link layer connection should also verify its IP/Locator configuration. If it is not configured yet, an initial configuration has to be installed. Otherwise, if it is already configured, it must be ensured that the

current configuration is consistent with the network the interface is connected to. In case of inconsistencies, the configuration has to be adapted. Therefore, the Base Station/L2 configuration component should trigger the IP/Locator configuration component after configuring a link layer connection.

2) After configuring or updating the configuration of an interface, the IP/Locator configuration component should notify the Router configuration component. (Re)-Configuration of an IP/Locator might force a router to adapt its address space allocations, e.g. if the newly configured IP/Locator is not in its currently managed address space. Furthermore, if the new configuration is caused by a newly available connection, this connection might provide additional information, e.g. about other AN domains. Using this information, the Router configuration component is able or possibly even forced to adapt or optimise its configuration.

3) The routing configuration component should constantly verify and optimise its configuration. This is even more important in case of topology changes. As most router configuration changes result in topology changes, router configuration components should propagate topology changes to the routing configuration component. As the routing tables are updated, the peer management database should be adjusted, as well.

The interactions described above basically define the bottom-up information flow when a new element enters an AN. However, there is also the possibility that top-down interactions have to be performed within an already configured AN. Additional interfaces for component interactions are required for a) routing optimisation process might be initialised to assure optimal routing, b) policy optimisation for routing configuration components. Detail descriptions can be found in [3].

4.4 Traffic Engineering (TE) Approach

For a network operator it is important to tune the performance of the network in order to optimise the usage of network resources. This is often referred to as Traffic Engineering (TE). The objective is to accommodate more traffic with the same amount of resources. Complementary to TE, but acting on a much longer time-scale, is network planning. A carefully planned network needs less traffic engineering. However, future communication networks such as AN, are expected to accommodate more mobile users leading to shifting traffic demands and in some cases a more dynamic network topology. This will make the task of network planning more difficult or in some cases impossible. Consequently, TE will be even more important in the operation and management of AN. In addition, for future mobile communication networks, business issues must be taken into consideration in providing connectivity anywhere, anytime at a low cost. Efficient resource usage is the key objective of TE. Hence, traffic engineering will play an important role in future mobile communication networks, particularly wireless access networks.

As previously stated, TE mechanisms dynamically adapt packet forwarding based on the state of the network. To know the state of the network, some network level parameters (link loads, packet losses, etc.) have to be monitored in some way. For instance, every router can monitor its output interfaces and store the obtained values in MIBs. Periodically, TE mechanism will retrieve this information. Depending on the type of mechanism, different ways to manage this information are used:

- If a centralised mechanism is used, routers will send the required information (e.g. load of every link in the network) to the entity running the optimisation, which will manage global information on the state of the network.
- If the mechanism is decentralised (running in each router) and needs global information of the state of the network (e.g. load of every link in the network), routers will send values to each other (e.g. using the IGP flooding functionality).
- If the mechanism is decentralised and it only needs local information (e.g. average load of output interfaces of a router), no information need be sent.

The first step in the traffic engineering process is to collect sufficient information to optimise the routing in the network. This information can be succinctly captured in a traffic matrix that has one entry for each origin-to-destination demand. The traffic matrix together with routing information is instrumental in traffic engineering. Despite the importance of knowing the traffic matrix, the support in routers to measure it is poor and operators are often forced to estimate it from other available data. An often-suggested method is to use link loads together with routing configuration to estimate the traffic demands. The link loads are readily obtained from the Management Information Base (MIB) of the routers using the Simple Network Management Protocol (SNMP). However, since the number of links is typically much lower than the number of node pairs in a network, the problem is highly under-constrained. In order to add more constraints to the estimation more information must be added. This information is usually some assumption made about the behaviour of the traffic demands. In the literature there are several suggested assumptions made to address the problem. For a balanced evaluation of a wide selection of estimation methods see Gunnar et al [15]. The evaluation of traffic matrix estimation methods in [15] was solely focused on the error in the estimation. However, the error in itself has little value unless the results are related to an application of the traffic matrix. This is one of the issues addressed in this study. How accurately must the traffic demands be known in order to have robust and efficient routing in the network. We will study how search heuristics perform when subjected to estimated as well as dynamic traffic demands and emerging hot-spots, i.e. sudden large traffic demands appearing in limited areas of the network. A related issue is the time scale for the estimation. In order for the estimation to be useful, it must be a good prediction of the near future. That is, the properties that are estimated must be stable over some time horizon. This horizon must be longer than the time needed to perform the estimates. Otherwise the network conditions might have changed before the measurement estimates have been used.

The second step in the traffic engineering process is optimisation of the routing configuration. With the traffic demands as input, it is possible to find a routing configuration using either the MCF formulation previously mentioned or via search heuristics. MCF optimisation problems easily become large with tens of thousands of variables and constraints. But it is possible to calculate the global optimal solution in tens of seconds even for large networks [16] if no constraints are given on the number of paths that can be used. Finding the optimal set of weights in OSPF, though, usually has to rely on heuristic methods. Implementing an optimal solution in the network would require extensions to the current IP routing protocols or the use of more sophisticated forwarding mechanisms such as MPLS (Multi Protocol Label Switching).

Using search heuristics to find link weights, on the other hand, can easily be combined with legacy routing protocols and forwarding equipment, for instance to find an efficient weight setting in OSPF or IS-IS routing. The drawback of this approach is that the solution found by the heuristic is not always the globally optimal solution. However, recent studies of the problem [17][18] have shown that in many cases, for instance for the topology and traffic demands in AT&T's global backbone network, the performance of the search heuristic is comparable with the optimal routing using the MCF formulation and MPLS forwarding.

Traffic engineering is a difficult task in regular IP networks and is even more challenging in Ambient Networks. On the other hand, the task of planning the network is also much more challenging in Ambient Networks due to the expected dynamics of topology and traffic patterns. Hence, the network must be able to autonomously adapt to the dynamics in the traffic situation and optimise the resource usage. This effort addresses the traffic matrix estimation problem and how errors in the estimation affect the performance of traffic engineering. Other steps in the traffic engineering process are addressed in [3].

5 Conclusion and Future Work

This paper discusses relationships between the concepts of autonomous and self-manageability and those of ambient networking, and the challenges and benefits that arise from their employment. Ambient Network research aims to enable the co-operation of heterogeneous networks, on demand and transparently, to the potential users, without the need for pre-configuration or offline negotiation between network operators. An integrated control space known as ACS is needed in AN for control purposes. An ACS contains different control elements for different aspects of ANs such as connectivity, mobility, media routing, context-awareness support, security and management. Four AN management applications and complementary approaches addressing the identified management challenges i.e. Peer-to-Peer, pattern-based, Plug-and-Play, and Traffic Engineering are analysed and fully described. Two strands of future work can be identified. Firstly the four management approaches will be continually refined and evaluated in synchronisation with the detailed technical work in the rest of the project. An integrated management approach based on the four approaches is under development in the next stage of the project. Secondly the Management component function interactions within the ACS will be evaluated and refined.

Acknowledgements

This paper describes work undertaken in the context of the Ambient Networks - IST project [1], which is partially funded by the Commission of the European Union. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the Ambient Networks Project.

References

1. WWI-AN Ambient Networks Project WWW Server-www.ambient-networks.org
2. Niebert, N., Flinck, H. Hancock, R. Karl, H. Prehofer, C.-"Ambient Networks – Research for Communication Networks Beyond 3G"-13th IST Mobile and Wireless Communications-Summit 2004, 27-30 June 2004, Lyon, www.mobilesummit2004.org

3. Galis, A., et al- “Ambient Network Management – Technologies and Strategies”, Report D8.1, http://www.ambient-networks.org/publications/D8-1_PU.pdf
4. Masahiko Matsushita- "Telecomm. Manag. Network", Vol. 3 No. 4, July 1991, page 117 – 122
5. CCITT -"Recommendation M.30, Principles for a Telecomm. Manag. Network", Geneva 1989
6. CCITT-"Recommendation M.3010, Principles for a Telecomm. Manag. Network", Geneva 1992
7. CCITT-"Recommendation M.3010, Principles for a Telecomm. Manag. Network", Geneva 1996
8. Baldi, M., Gai, S., Picco, G. - “Exploiting Code Mobility in Decentralized and Flexible Network Management”, First International Workshop on Mobile Agents, Berlin, April 97, pp. 13-26.
9. Baldi, M., Picco, G. -“Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications”, First International Working Conference on Active Networks (IWAN’99), June 1999, Berlin, Germany.
10. Liotta, G. Knight, G. Pavlou-“On the Performance and Scalability of Decentralized Monitoring Using Mobile Agents”, DSOM ‘99, Zurich, Switzerland, October 1999.
11. Lim, K.S., Stadler, R.- "Developing pattern-based management programs”, 4th IFIP/IEEE International Conference on Management of Multimedia and Network Services (MMNS’01), Chicago, Illinois, October/November 2001, pp. 345-358.
12. Adam, C., Stadler, R. -“Patterns for Routing and Self-Stabilization”, in Proc. of Network Operations & Management Symposium (NOMS 2004), Seoul, Korea, April 19.23, 2004
13. Schütz, S. (editor), “(Un)Plug-and-Play Architectures and Interfaces for Ambient Network Components”, Ambient Networks Internal Report, R8-5, November 2004.
14. Schmid, S.(editor), SMART – Draft Architecture and Multimedia Routing Decision Logic, Ambient Networks Deliverable D5-1, December 2004.
15. Gunnar, A., Johansson, M., Telkamp, T.-"Traffic Matrix Estimation for a Global IP Backbone – A Comparison on Real Data", In proceedings of IMC 2004, October 2004, Taormina Italy
16. Abrahamsson, H., Alonso, J., Ahlgren, B., Andersson, A., Kreuger, P.-"A Multi Path Routing Algorithm for IP Networks Based on Flow Optimisation", In Third COST 263 International Workshop on Quality of Future Internet Services, QoFIS, 2002.
17. Fortz, B., Thorup, M. -Internet Traffic Engineering by Optimizing OSPF Weights”, INFOCOMM 2000, March 2000, Tel Aviv Israel
18. Fortz, B., Thorup, M. -“Optimizing OSPF/IS-IS Weights in a Changing World”, Journal on Selected Areas in Communications, Vol. 20 no. 4, pp. 756-767, 2002.
19. De Meer, H., Tutschku, K., and Tran-Gia, P.-“Dynamic Operation in Peer-to-Peer Overlay Networks,” *Praxis der Informationsverarbeitung und Kommunikation, (PIK Journal),* Special Issue on Peer-to-Peer Systems, June 2003
20. Simon, C. et al,-“P2P Management in Ambient Networks”, IST Mobile Summit 2005.
21. Raz, D. -“An Active Network Approach for Efficient Network Management”, IWAN 99-International Working Conference on Active Networking; Berlin, Germany, June 30-July 2, 1999, Lecture Notes in Computer Science 1653 Springer 1999, ISBN 3-540-66238-3
22. Brunner, M., Galis, A., Cheng, L., Colás, J. A., Ahlgren, B., Gunnar, A. Abrahamsson, H., Szabo, R., Csaba, S., Nielsen, J., Prieto, A. G., Stadler, R., Molnar, G. – “Ambient Networks Management Challenges and Approaches”– invited paper ISBN 3-540-23423-3, IEEE MATA 04 1st International Workshop on Mobility Aware Technologies and Applications; 20-22 October 2004, Florianopolis, Brazil-www.ic.unicamp.br/mata04/
23. Galis, A., Denazis, S., Brou, C., Klein, C. (ed) – “Programmable Networks for IP Service Deployment” ISBN 1-58053-745-6; pp450, June 2004; Artech House Books; www.artechhouse.com

A Context-Aware Negotiation Model for M-Commerce

Fernando Menezes Matos and Edmundo R.M. Madeira

IC - Institute of Computing,
UNICAMP - University of Campinas,
13083-970 Campinas, SP, Brazil
{fernando.matos, edmundo}@ic.unicamp.br

Abstract. The low prices and the friendly usage of the mobile devices are contributing to increase the number of people who are using them for several tasks, as to pay a bill in an Internet banking, purchase a product, connect to a close printer and so on. Due to this, it is necessary develop applications which can cope with these new user habits. This paper proposes a context-aware negotiation model for mobile commerce between two participants, using mobile agents and taking into account the mobile device personalization. The model copes with the context-aware concepts through the use of created profiles. With these profiles it is possible to automate some tasks depending on the context, as for example the user identity.

Keywords: Context-Aware, M-commerce, Mobile Agents, Automated Negotiation.

1 Introduction

With the growth of mobile devices, a change in people's needs had come. This change is related to the fact that users can be anywhere while using their mobile devices to perform tasks and use services anytime.

In the next few years, these mobile services are expected to grow significantly in popularity, variety and complexity. But in order to avoid increasing complexity, and allow users to concentrate on tasks they want to accomplish, mobile services must be aware of their context and automatically adapt to the changing context, such as the place where the user is at, the time the application is used and even the user profile [7]. Due to this, the development of context-aware applications is becoming ever so important.

Mobile commerce, or "m-commerce", is another activity which has resulted from the growth in use of mobile devices. Mobile commerce is a type of electronic commerce where a user can purchase a product or service no matter where she is at. However, to do this successfully, it is necessary to develop techniques and communication mechanisms for mobile commerce, which satisfy the requirements and the limitations of the mobile infrastructure, like the reduced screen size of the cellular phones and the low bandwidth of the wireless networks [8].

There are already many applications that cater to these limitations, even so, there are still many opportunities for providing more sophisticated, intelligent services to users on the move. Such services should go beyond basic information provisioning and organisation tasks by providing intelligent and pro-active support to mobile users, before and during the execution of their tasks [6].

An issue to considering in mobile commerce is the fact that mobile devices are personal. Therefore, the development and usage of user profiles is of interest. Performing transactions using a user profile would be more similar to what is done in real life. This would result in guaranteeing great purchase personalization

This paper proposes a context-aware negotiation model for mobile commerce using mobile agents based on a preexisting negotiation model for mobile commerce [4]. It also shows how this model can be extended to cope with context-aware issues using the user profile.

Section 2 presents the proposed model with its operation and phases. Section 3 presents some implementation aspects, including some measures. Finally, Section 4 presents a conclusion and lists future works.

2 The Negotiation Model For Mobile Commerce

2.1 Proposed Context-Aware Model

This Section presents a context-aware negotiation model for mobile commerce where buy agents and sell agents negotiate between themselves to achieve the best deal. This negotiation is based on the price and the product features. The price negotiation is based on the Kasbah model [1].

Figure 1 presents the state diagram of the proposed negotiation model. This model is based on the proposed OMG bilateral negotiation model, where two parties negotiate to achieve a mutual advantageous deal. The negotiation in the OMG model may be initialized under one of three states: *proposed*, *requested* or *offered*. *Offered* implies a state in which the negotiation subject may not be changed. The *proposed* state allows the subject negotiation to be changed to the *requested* state. The *requested* state allows a participant to reach the *offered* or *proposed* states.

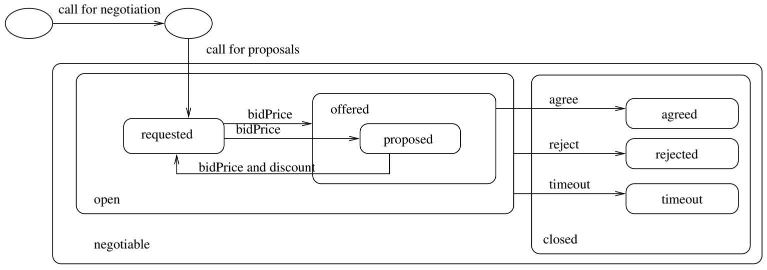


Fig. 1. Negotiation Model extended for Mobile Commerce

Our model differs from the OMG in some aspects. One difference is in an initial transition, called *callForProposals*. This transition substitutes all three initial transitions of the OMG model, initiating by the *requested* state. Another difference is the creation of a new message called *callForNegotiation*. This message is used to express the users desire to start a new negotiation. There is also the presence of a new message called *discount*. This message is used only once, after the participants have reached an agreement, as the buy agent final counter-proposal to the sell agent. It is used in the case the sell agent uses a fidelity program and wants to give a discount to the client. The last change was the creation of the *bidPrice* transition. This transition replaced the *offer*, *propose* and *request* transitions, that were the transitions used in the three *open* states (*requested*, *offered* and *proposed*). The *bidPrice* value will define in which state the model is at the moment. A negotiation is finished when any sub-state in the *closed* state is reached.

The model works as follows. The client creates an agent called original buy agent, which contains the desired product features and the maximum price to pay. This agent replicates itself, so that several instances of the same agent, called buy agents, can negotiate with several sellers simultaneously. Each seller has a sell agent, to negotiate with a buy agent, and it does so based on a pre-defined price and features. The proportion between sell agents and buy agents is one to one. The sell agents have catalogs where there is information about their products. The catalogs are implemented in XML to provide interoperability between different agent systems. After discarding some options, due to the lack of the product or some feature, the buy agents come back, with the offers, to the original buy agent, which sends the offers to the client, who decides on the purchase.

The desired product features define which sell agents the buy agents will negotiate with. When the user defines the desired features, she also defines a weight which varies from 0 to 10. The greater the weight is, the more important the feature is. After that, the product total weight is calculated, which is the sum of the squares of the weight of each feature. The weights are squared to emphasize the more important features defined by the user.

The similarity defines how similar a product is to another based on the total weight of its features. When a sell agent performs a search in its catalog to find the desired product, it verifies which features the found product has. For each found feature, the same weight defined by the user is assigned to the seller product, and its total weight is calculated. If the total weight of the seller product is greater or equal than the total weight of the desired product, this seller is chosen by the original buy agent to negotiate. We also use the similarity concept to perform the purchase.

2.2 Context-Aware Issues

Our model was designed to cope with a challenge to exploit the environment changes that occur with the users of mobile devices [5]. These changes require new classes of applications that are aware of the context in which they run.

Context is any information that can be used to characterize the situation of an entity [3]. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves.

A context-aware application is one in which it has knowledge of its physical surrounding and computing environment. This environment is composed of people, mobile and fixed computing devices, and things such as doors, walls and rooms. Such context-aware software adapts according to the location of usage, the collection of nearby people, hosts and accessible devices, as well as to changes to the environment. Almost any information available at the time of an interaction can be seen as context information. Chen and Kotz in [2] divide context into four categories: *Computing context*, such as communication bandwidth, communication interference and nearby resources, such as printers and workstations; *Physical context*, such as noise levels, temperature and light; *User context*, such as the user's profile; and *Time context*, such as time of the day, week and month.

Our model satisfies the user context, since it uses the user and purchase profiles, during the negotiation, to personalize the user's transaction. It also satisfies a location context, which can be a physical context type, since through the profiles, the user can obtain advantages when she enters in certain places, like stores.

One of the inherent features of mobile devices is their customization. Hence, the creation of a user profile to be used in mobile commerce is important. Besides, they can be used as identity information for context-aware applications. In this work, we have created two types of profiles. The first one is related to the user. Each item of this profile influences somehow the negotiation and the items are not often altered. Once they have been defined they are valid for every transaction the user makes. This profile contains the following items:

- *Number of Negotiations*
  This item defines the maximum number of performed negotiations in a transaction, i.e. the maximum number of buy agents copies created by the original buy agent to negotiate.
- *Number of Results*
  This item defines the maximum number of desired responses, i.e. how many responses will return to the user.
- *Automatic Purchase*
  This item defines whether the original buy agent must perform the purchase after all results have returned, without previous consultation to the user. With this item the user delegates to the original buy agent the responsibility to perform the purchase on her behalf, based on its context information (profile and negotiation results).

The other profile type is the purchase profile. In this profile, the user can set the values of the items that compose the profile for each transaction she performs. This profile contains the following items:

- *Negotiation Weight*
  This item defines whether the weight of the product features will be taken into account when choosing the product. If it is not enabled, only its price will be considered.
- *Fast Negotiation*
  If this item is enabled, the original buy agent performs the purchase of the first incoming result below the desired price, without informing the user about the negotiation results, and informs all the other buy agents to stop the negotiation.
- *Fidelity*
  This item defines if the user wants to keep fidelity to a seller. The original buy agent performs the purchase of the first incoming result, if its price is lower than the desired price and if it is a result from that specific seller. Otherwise, it returns all results to the user to decide. This item can be used also when the user enters the seller's store that she keeps fidelity. Automatically, a discount would be granted to the user in a specific or in all products. The context-aware based application can easily be seen in this scenario.
- *Connected*
  After recording, the user is disconnected. The next time the user connects, pending negotiations are checked. If any exists, the connection is established with the original buy agent and the results are requested.

Besides the usage of the profile items, it is possible to do a record of the user actions during a negotiation, obtaining a profile of the user purchases and making a record of her preferences. This allows, for instance, to send messages to the user, when a product similar to which the user made a negotiation sometime is in special offer. The application can also show to the user if a store where she has already bought something is near to her. Thus she can go to the store and look if it has other products of interest or similar to the one she bought.

2.3 Phases

The model for mobile commerce is divided into 7 phases. Next we will explain each phase. Figure 2 shows the sequence diagram of the model with the message flow between the model modules. The full arrows represent the model messages. The dashed arrows represent the messages of the mobile agent platform. The dotted arrows represent the messages which do not change the state of the model. The numbers in parenthesis are used as references to the messages in the following text.

- Connection Phase
  After the user defines the features of the desired product, the application *BuyApplication*, which is in the mobile device, sends a message *callForNegotiation*(1) to *CallApplication*, via *Proxy*, which are modules in the fixed network. *CallApplication*, considering the product information, creates (2) an original buy agent (*BuyAgent*) and saves (3) the information of the negotiation session in *PendentNeg*. The original buy agent is created with the

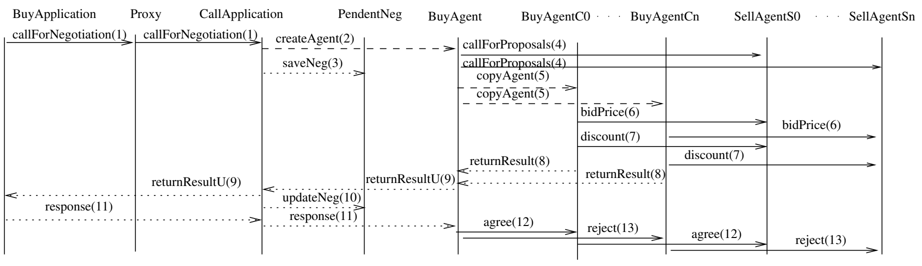


Fig. 2. Connection sequence diagram

following information: product features and respective weights, the maximum price and the price the user would like to pay for the product and the user and purchase profiles.

- Call Phase
  In this phase, the original buy agent starts a search for sellers which have the user's desired product, using the message *callForProposals* (4). This search takes into account the product features and their weights.
- Selection Phase
  In this phase, the original buy agent selects the sell agents. This selection is performed among the sell agents which affirmatively responded in the prior phase, taken into account the weight of the product in the seller and the *Number of Negotiation* item of the user profile. In this phase there are not message exchanges.
- Negotiation Phase
  After the sell agents selection, the original buy agent creates copies of itself (5) and sends them to the hosts of those selected sell agents. When they arrive in the sellers hosts, the buy agent copies start the negotiation with the sell agents. This negotiation is based on the maximum and minimum prices defined by the user and the maximum and minimum prices available in the seller. This is done using proposals and counter-proposals between the agents. All this communication is performed by the *bidPrice* message (6). If an agreement occurs, the buy agent still asks for a discount (7) to the sell agent. The sell agent may grant the discount or not, depending on the context, for example, whether the user is its client or not.
  At the end of this phase, results are returned to the original buy agent (8).
- Analysis Phase
  In this phase, the original buy agent analyzes the returned results. This analysis is performed considering the contextual information of the negotiation, in other words, the user and purchase profiles. If the *Fast Negotiation* item is enabled, the incoming responses are immediately analyzed by the original buy agent. When the first result is returned, from a buy agent, in which the obtained price is lower than the desired price, the original buy agent performs the purchase and informs all other buy agents to stop the negotiation. In case the *Fidelity* item is enabled, when the first result is returned, in which the

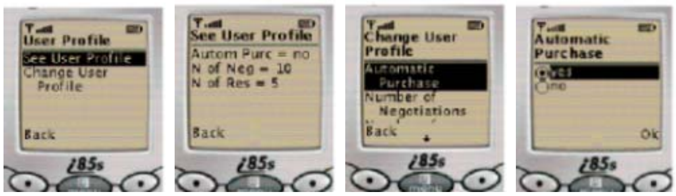
obtained price is lower than the desired price and the result came from a seller the user wishes to keep fidelity to, the original buy agent performs the purchase. If the *Negotiation Weight* item is enabled, the original buy agent waits the return of all results and it sorts them by a relative weight. This is obtained by dividing the product weight by its price (both values were returned by each buy agent). This is done if the client wants to take into account the product features when performing the purchase, since a seller is only selected if the weight of the product in its catalog is greater than, or equal to, half of the weight the user has defined. This prevents the client from buying the cheapest product, whose features are far from her requirements. Finally if the *Automatic Purchase* item is enabled, the original buy agent waits the return of all responses and then it asks for the copy that got the smallest price to perform the purchase.

– Presentation Phase
  Once the original buy agent has already taken all the decisions about the returned proposals, it sends the results (9) to the mobile device to be presented to the user. After that, *CallApplication* informs *PendentNeg* (10) about the negotiation situation. In case of a connected negotiation, defined by the *Connected* item of the purchase profile, *PendentNeg* is warned to discard the information it has about the negotiation, once the results were returned to the user. Otherwise, if a disconnection happened, due to the user decision, or a failure environment, *PendentNeg* saves the negotiation results to be presented to the user when she requests them.
– Conclusion Phase
  In this phase the user can choose, among the returned results, one which satisfies her, sending a response (11) to the original buy agent. With the user response, the original buy agent accepts the chosen seller (12) and rejects the other sellers.

3 Implementation Aspects

A prototype was developed to validate the negotiation model. The agents were implemented using the Java 2 Standard Development Kit 1.4.0 (J2SDK 1.4.0) and the application at the mobile device was implemented using the version 1.0.4 of the Java 2 Micro Edition Wireless Toolkit (J2ME). Grasshopper version 2.2.4 was the agent system used to supply the communication infrastructure. It was chosen because its interoperability with different agent systems due to its MASIF (Mobile Agent System Interoperability Facility) interface support. The catalogs were implemented in XML.

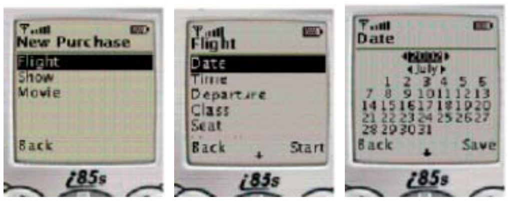
Figure 3 shows some screenshots when the user chooses the *User Profile* option from a prior screen. In the screenshot of Figure 3(a), it is possible to visualize or change the profile, while Figure 3(b) presents the user profile screenshot. In the screenshot of the Figure 3(c) the user chooses which profile attribute she wants to change. Finally, in the screenshot of Figure 3(d) the user can change the chosen attribute.



Autom Purc = no
N of Neg = 10
N of Res = 5

(a) (b) (c) (d)

Fig. 3. Managing user profile screens



(a) (b) (c)

Fig. 4. New negotiation screens

Figure 4 presents some screenshots which are used to perform a new negotiation. Figure 4(a) presents the screenshot where the user chooses the desired product, a flight ticket in that case. In Figure 4(b), the user chooses some flight feature, while Figure 4(c) shows the user defining the date of the flight.

One challenge of developing a prototype for mobile devices is the limited amount of memory. Due to its small size, the devices do not support applications with great memory consumption. However, with the development reached by electronics, it is possible to find devices with more memory capacity. The new cell phones, like Motorola E398 and Nokia 6680, have at least 2M of memory, which allows them to run more complex applications.

Our application was developed to cope with the limitations of the mobile infrastructure. It was developed as small as possible in memory consumption and source sizes. Table 1 shows the maximum memory consumption of the prototype in the mobile device. As we can see, the maximum memory consumptions were far below the total memory available of the cell phones. Table 2 shows the memory consuming after the garbage collector execution.

Table 1. Measures before garbage collector

max memory used	total memory available	free memory	objects in the heap
105K	235k	181k	1137

Table 2. Measures after garbage collector

current memory used	total memory available	free memory	objects in the heap
20K	235k	215k	468

Table 3. Time measures

action	time (in milliseconds)
call for proposals	379
copy of the agents / negotiation in parallel	448
copy of the agents / negotiation in sequence	409
negotiation	10

We took measures in two scenarios: negotiation in parallel and negotiation in sequence. The multicast message (*call for proposals*) and the negotiation of one agent in the two scenarios were the same. The difference was in the copy of the agents. The total time of the negotiation in the parallel scenario was $(379+448+10) = 837$ milliseconds. The total time in the sequence scenario was $379 + (n * 10) + (n * 409)$ milliseconds, where n is the number of sellers. Table 3 shows the measures.

In the proposed model, it is possible to download product definitions to customize the application. These definitions correspond to the features of the product and are used to change the application interface according to the product the user wants to negotiate. For instance, if the user wishes to buy a CD, she can connect to a server which makes the product definitions available, which are genre, title and artist. Thus, it is possible to perform transactions on several products since the definitions are available.

4 Conclusion

This paper proposes a context-aware negotiation model for m-commerce using mobile agents. This model can be used to develop applications in which a user wants to make a purchase. The model takes into account the characteristics of the wireless environment, such as the low bandwidth of the wireless network, the low processing power of the mobile devices and the fact these mobile devices are personal. User and purchase profiles were defined, which guarantee the purchase personalization. The profile items form a context, which the original buy agent uses to make decisions about the purchase, thus creating an agent with more power decision. The model supports user disconnection, in other words, in the case of a disconnection, the user can request the negotiation results in a new connection. All the heavy computation is performed in the fixed network, by the *CallApplication* and the agent platform, and it does not overload the mobile device, and does not overload the communication channel, because it imposes small data traffic between the mobile device and the fixed network as well. A flight ticket booking application was developed to validate the model.

As future work we intend to define ways of payment, delivery details and warranty to be used in the transaction, as well as to extend the model to take into account these features during the product negotiation. We also intend to define an expiration date to the offers. This expiration date will be used as a mechanism that allows the seller to force the original buy agent to perform the purchase soon. The offer would have a lower price during a short period.

References

1. Anthony Chavez and Pattie Maes. Kasbah: An agent marketplace for buying and selling goods. In *First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96)*, pages 75–90, London, England, April 1996.
2. Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.
3. M Korkea-aho. Context-Aware Applications Survey. http://www.hut.fi/ mkorkeaa/doc/context-aware.html.
4. Fernando M. Matos and Edmundo R. M. Madeira. An automated negotiation model for m-commerce using mobile agents. In *Proceedings of the International Conference on Web Engineering (ICWE 2003)*, volume 2722 of *Lecture Notes in Computer Science*, pages 72–75, Oviedo, Asturias, Spain, July 2003. Springer-Verlag.
5. Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.
6. Iyad Rahwan Talal Rahwan, Tarek Rahwan and Ronald Ashri. Agent-based support for mobile users using agentspeak(l). In *Agent-Oriented Information Systems AAMAS-04 Workshop*, volume 3030 of *Lecture Notes in Artificial Intelligence*, pages 47–62, New York, USA, July 2004. Springer-Verlag.
7. Da Qing Zhang Tao Gu, Hung Keng Pung and Xiao Hanh Wang. A middleware for building context-aware mobile services. In *Proceedings of IEEE Vehicular Technology Conference (VTC 2004)*, Milan, Italy, May 2004.
8. Aphrodite Tsalgatidou, Jari Veijalainen, and Evaggelia Pitoura. Challenges in mobile electronic commerce. In *Proceedings of Iec 2000. Third Internacional Conference on Innovation through E-Commerce*, pages 169–174, Manchester, England, November 2000.

Collection and Object Synchronization Based on Context Information

Sasu Tarkoma, Tancred Lindholm, and Jaakko Kangasharju

Helsinki Institute for Information Technology,
P.O. Box 9800, FIN-02015 HUT, Finland
{sasu.tarkoma, tancred.lindholm, jkangash}@hiit.fi

Abstract. We present a novel mechanism for collection and object synchronization based on context information. The mechanism is based on a distributed event system and uses event filters to represent context and realize context queries. The central operations of the system are storing and retrieving objects by their context. The new feature of the system is context-based synchronization, which allows synchronizing collections of objects continuously based on the given context. The system may also be used for context-based service provisioning. We present mechanisms for both collection and object synchronization. The former uses the publish/subscribe paradigm and the latter builds on an XML-aware file synchronizer. We focus on the first mechanism and also discuss the use of the system on mobile and wireless devices. We present a context-aware photo library as an example application.

Keywords: Context-awareness, data synchronization, Publish/ Subscribe.

1 Introduction

A number of core technologies are needed in order to realize the intelligent and adaptive services of tomorrow. Efficient and intelligent *data synchronization* is a basic property of current and future applications, especially in mobile and ubiquitous environments. Mobile phones, laptops, and PDAs have become commonplace. We are faced with the question of how to locate important data items and keep them synchronized on different devices. Furthermore, the rules for synchronization are dependent not only on the device, but also on the current and future operating context. In this paper, we present a *middleware* system and an *API* for creating and tracking object collections based on *context queries*, and then synchronizing the objects using a file *synchronizer*.

Context-awareness is considered as an important property of future mobile applications [1]. In this paper, context is represented by a set of dimensions that take either discrete or interval values. We focus on how context information may be used to synchronize objects and do not consider how the actual context information is acquired.

The proposed mechanism is based on three basic middleware services included in the Fuego middleware service set: the *messaging service* based on synchronous and asynchronous SOAP [2], the *event service* [3] that facilitates distributed *publish/subscribe* (pub/sub), and the *XML-aware file synchronizer* [4]. The messaging service is responsible for transporting information between known entities using for example explicit addresses or queue-names. The event service is responsible for decoupled, anonymous one-to-many information dissemination [5]. The XML-aware file synchronizer provides facilities for the synchronization of files and directories using XML directory trees and a tree reconciliation mechanism.

The contributions of this paper are: 1. Using techniques from publish/ subscribe systems to match and compare context information. This allows both point and subspace matching in the context/content space. 2. An API for tracking and synchronizing collections using context queries. 3. Using the distributed pub/sub to synchronize collections, and 4. using an XML-aware synchronizer to synchronize files. The system may be also used for context-based service provisioning and context-based personalization of applications.

This paper is structured as follows: in Section 2 we discuss how context information may be represented using event filters. Section 3 presents the proposed synchronization system and gives examples of context-based synchronization. In Section 4 we examine the example application and Section 5 discusses related work. Finally, Section 6 presents the conclusions.

2 Representing Context with Filters

Event systems [5], such as Siena [6], Rebeca [7], and Elvin [8], support content-based routing of information, in which events are delivered to subscribers based on their content. Content-based event routing has been proposed as one of the requirements for advanced applications, in particular for mobile users [9, 10]. Publish/subscribe has also been proposed for metadata management in distributed environments [11]. The content-based address of a client is dynamic and may be used to realize context-sensitive messaging. Context-sensitive messaging using content-based routing is implemented by subscribing the current context, after which all events matching the context will be delivered to the subscriber. The context is represented using a filter. The expressiveness of the subscription language is one of the main factors that define the power and scalability of the event service.

Filters are used in pub/sub systems to specify user interests and make routing decisions. We follow the general notification data model: a notification is a set of 3-tuples <name, type, value>. A subscription contains a filter, which is a set of attribute filters. Attribute filters are constraints on tuples in notifications. Therefore, a notification is a point in the content (or context) space, and a filter is a point or a subspace.

Filters typically have two useful properties: *covering* and *overlapping*. Filter F_1 is said to cover filter F_2 if and only if all the notifications that are matched by F_2 are also matched by F_1. Overlap is defined similarly and happens when the two filters match the same arbitrary notification. If filters are organized in a graph based on the covering relation, several optimizations are possible for matching filters against notifications. For example, if filter F_1 does not match a notification, there is no need to test those filters that are covered by F_1.

Typically, event systems manage filters simply as sets. The Siena system uses a data structure called the *filters poset* (partially ordered set) to maintain routing state between content-based routers. The poset determines and stores the *covering* and *covered* filters for each input filter. A filter may be associated with several output interfaces. We have specified and implemented a new data structure called the *poset-derived forest*, which stores only a subset of the relations to optimize filter processing and support frequent updates. Experimental results indicate that the forest is considerably more efficient than an acyclic graph-based poset under frequent updates [12]. We have developed a graphical tool, called the *Poset-Browser*[1], for experimenting with various content-based routing data structures.

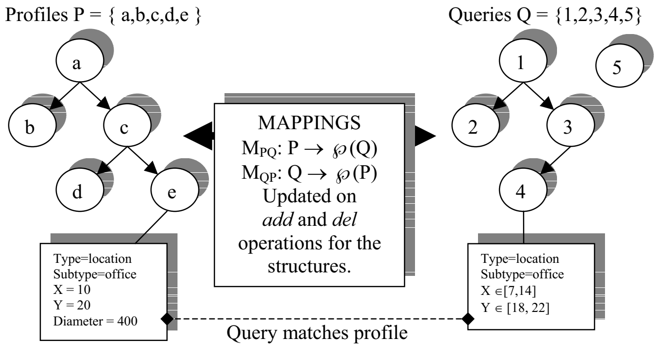


Fig. 1. Storing and matching of profiles and queries

A generic data structure, such as the poset-derived forest, supports the use of arbitrary filter objects as long as the covering relations are defined for the input set. On the other hand, being generic they cannot provide the same performance as filter object and language specific matchers. It is more efficient to use a separate data structure for matching. In general, filter matching is done by counting attributes using the counting algorithm [13, 14] or using a tree-based data structure [15]. These more efficient matching mechanisms assume that the filters and notifications have a distinct and often simple structure. Moreover, typically the most efficient matchers do not support frequent changes in the set of queries, which poses challenges for context queries.

We propose that the poset-derived forest can be used to store both context profiles and context queries based on the covering relation. The forest supports frequent updates to the data structure and it has approximately the same matching performance as the poset. Figure 1 illustrates how two forests may be combined to support the matching of profiles and queries with mappings between

[1] Available at www.hiit.fi/fuego/fc/demos

the elements of each forest. The basic idea is of this new data structure, called DoubleForest, is to maintain mappings M_{PQ} and M_{QP} in different directions between the structures. M_{PQ} determines the set of covering queries given a profile, and similarly, M_{QP} determines the set of covered profiles given a query. It is easy to produce *change notifications* to relevant entities when an element is added or removed from either structure. This approach is general and supports various filtering languages; however, it does not work well if there are only a few covering relations between the elements.

We have developed an online example of this mechanism called the *ContextBrowser*[1]The ContextBrowser shows context queries and profiles graphically and demonstrates real-time collection tracking. Initial results with the DoubleForest data structure for context matching indicate significant performance benefits compared to naive set based matching.

3 Synchronizing Collections

We follow the ideas in [8] and propose to support basic communication between context providers and consumers using a pub/sub event routing network. In addition, we propose to leverage the covering and overlapping relations between filters to realize context-based object synchronization. The separation of concerns offered by pub/sub systems simplifies the development of higher level components, because mobility transparency and scalability is handled by the lower pub/sub layer.

Figure 2 presents an overview of the synchronization process. The environment consists of a number of client systems (terminals) and a number of synchronization service instances. Consider a scenario, in which the client is interested in documents modified in specific project meetings. Relevant metadata and contextual information is added to documents after meetings so that this information is available. The client can formulate a context query that includes those specific meetings and track documents that match this query using the service. The pub/sub system delivers change notifications and the collection defined by the context query is updated in real-time. In collection synchronization the client receives notifications that denote changes to the collection, namely addition and removal of entries. At some point when tracking the collection, the client may be interested in retrieving the documents to his current machine. This is accomplished using the file synchronizer, which updates local directory structure and downloads the requested files. This motivates the different phases of the synchronization process.

Today security is also an important requirement and in many cases profiles can be matched to queries only when *access control rules (ACR)* allow it. ACRs can also be represented using filters. A valid context query is covered by at least one access control rule that is applicable for the client that made the query. Black hole queries may be enforced by using access control lists or by requiring that all queries conform to a schema. The schema may then be used to restrict the expressiveness of the query.

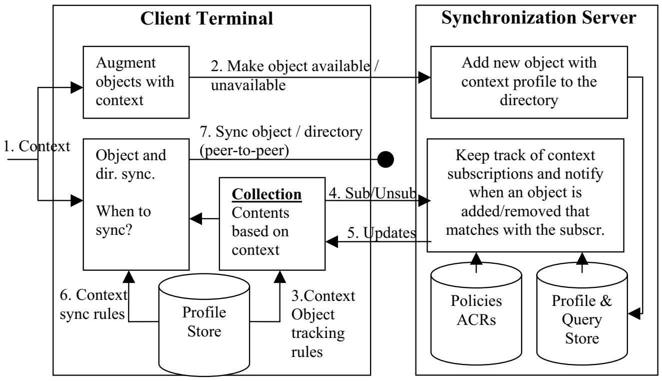

Client Terminal
Synchronization Server
Augment objects with context
2. Make object available / unavailable
Add new object with context profile to the directory
1. Context
7. Sync object / directory (peer-to-peer)
Object and dir. sync.
When to sync?
Collection
Contents based on context
4. Sub/Unsub
5. Updates
Keep track of context subscriptions and notify when an object is added/removed that matches with the subscr.
6. Context sync rules
Profile Store
3.Context Object tracking rules
Policies ACRs
Profile & Query Store

Fig. 2. Context-based collection and object synchronization

We assume that there is an existing mechanism for acquiring the current context of the client system (1). The current context is associated with objects, files, and directories that clients make available using the system (2). The synchronization service is responsible for maintaining objects and their associated context profiles, which are updated by clients. The client system stores two different kinds of filters: first, the context queries needed for tracking objects based on their context (3), and then a set of rules that are used to synchronize subsets of these tracked documents. The synchronization rules are motivated by the observation that the client may not always wish to synchronize the whole collection, but only a subset of it.

The service is also responsible for accepting collection tracking subscriptions and unsubscriptions from clients (4). The service keeps the client notified about the current contents of the tracked collection (5). The service uses the poset-derived forest data structure to represent context profiles, context queries, and access control rules.

The client uses a set of synchronization rules for synchronizing subsets of the tracked documents (6). A subset of the collection is synchronized using the file synchronizer (7). The synchronized objects can reside anywhere in the distributed system, for example on other terminals.

We divide the synchronization of objects and collections by their context into the following three parts:

– Description: How to describe context and how to describe objects. We separate the description of an object from the object itself. This allows the system to keep metadata separate from the actual location of the object. Typically the object description, denoted by O_D, contains a pointer (URI) to the object. The description also includes the context profile.
– Continuous collection synchronization: How to synchronize a collection defined by a context query. A query exactly defines the contents of a collection to be those profiles that match the query. The query may also contain meta-

data relevant for matching. If the change notifications are ordered using causal or total order the client can apply them in the correct sequence. The communication is performed using asynchronous publish/subscribe.
– Document and directory synchronization: How to synchronize the data associated with the items. We model the data as generic XML documents and use the file synchronizer to perform one-shot synchronization after periods of disconnection.

Collection synchronization differs from the basic pub/sub semantics, because context profiles have temporal duration, but events typically are instantaneous and published only once. This means that both profiles and queries must be maintained by the service and the mappings introduced earlier are needed between these sets. We define the *context-matching problem* from the viewpoint of change notification as follows:

Given a set of context descriptions and a set of context queries:

1. for a new profile, find the set of associated (matching) queries and their owners that need to be notified,
2. for a new query, find the set of associated (matching) profiles,
3. when removing an existing profile, notify the set of associated query owners of the removal.

The synchronization service is logically centralized and based on the client-server interaction model. We use the service to solve the context-matching problem. The service exports a pub/sub API for adding and removing object descriptions and context queries. Clients make requests and receive change notifications using the pub/sub system, which abstracts issues such as disconnections and mobility. The service is implemented using a context-matcher component that stores both context profiles and object descriptions. The component uses the underlying event system for communication and matching filters and notifications. The implementation of the component is simple, because existing data structures and communication primitives are used.

In addition to the client-server model, it is also possible to implement the system using the peer-to-peer model. The peer-to-peer model does not have this centralization of service instances and allows entities to produce and consume context information directly using pub/sub interfaces. This means that clients need to keep track of their objects and the associated context filters, and be able to match them against incoming queries. This approach requires more from clients, but does not require a centralized component. On the other hand, features such as authentication [16] and logging become more difficult to realize.

3.1 Operations

The central operations of the system are storing and retrieving objects by their context. Let F_D denote a context profile that represents the current or past context of an object. Table 1 presents the basic API operations. The API consists

Table 1. Basic API operations

Operation	Description
add(O_D,F_D)	Adds the context profile F_D to object desc. O_D.
del(O_D,F_D)	Removes the context profile F_D from object desc. O_D.
get(F)	Get object descriptions (one-shot) for the given context query F.
add(C,F)	Adds the context query F to the collection C.
del(C,F)	Removes the query F from the collection C.
sync(C)	Sync C using the synchronizer.

of two parts: addition and removal of objects and their descriptions, and the management of collections. The collection C is determined by a set of context queries. The API allows the client to add and remove these queries.

A client may modify or remove only objects that belong to it. The synchronization service must keep track of object descriptions sent to clients and only new descriptions are sent. Similarly, clients are notified when a change or deletion of a description changes their collection. The operations may be extended with access control rules, for example, rules for deciding which entities may access the object or receive a notification about it.

The context profile associated with an object (the object's context) may be a point or a subspace of the content space. Context queries associated with collections are assumed to be subspaces. We distinguish between matching points and subspaces, because there are efficient and sublinear algorithms for matching points [13, 14]. Algorithms for covering and overlapping are computationally more expensive. Covering and overlapping can be computed efficiently for the commonly used filtering model, in which each tuple in a filter contains exactly one predicate (constraint).

The *sync* operation creates local replicas for new objects, updates concurrent changes to distributed copies, and removes files that disappear from the collection. The URI may also map to a directory, in which case the whole directory is synchronized based on the given context. It may also happen that the client wishes to synchronize only a subset of the collection. This is supported by the API and we envisage that additional rules may be used to determine this subset.

3.2 Mapping to the Pub/Sub Paradigm

The collection synchronization service builds on the Fuego event system and maintains two data sets: the set of object descriptions and the set of context queries. The service subscribes to receive new object descriptions and context queries. The client on the other hand subscribes its event message queue name. This provides a transparent way to send messages to the client irrespective of physical location using the queue name. The service multicasts updates using the recipient queue names. This means that only a single update message is sent by the service. The pub/sub system then forwards the replicates the update message when necessary. The default Fuego filtering language supports the basic relational and string comparison operations. The default language may be extended or replaced to support more expressive languages.

One useful feature for collection synchronization is the *replace-identifier* feature of our event system, which allows one notification to replace other notifications in a message queue that have the same source-specified identifier. The replace-identifier in conjunction with *time-to-live* are useful in removing stale and obsolete messages from queues before they are delivered to clients, thus reducing wasted bandwidth and client-side processing. For example, they may be used to ensure that only one notification is sent for a changed object.

The *add*, *del*, and *get* client collection API operations are implemented using the publish operation of the event system. They synchronization service must subscribe these operations before they may be used.

Scalability may be improved by separating orthogonal context types (or schemas) into different distributed service modules. It is assumed that a single service instance is responsible for a subspace of the context space and can thus uniquely number collection change notifications for that subspace. If this design is used, the clients must treat change notifications for the same object from different service instances as independent. This means that a remove notification from one service instance does not remove the object from the client's collection if the client has previously received an addition for the same object from a different service instance.

3.3 Sequence Diagram

Figure 3 presents a sequence diagram of synchronization with pub/sub operations omitted. In 1. the client B stores the object O using any available storage mechanism and either receives or formulates a URI for the object (2). After this the client creates the description O_D of the object that contains the URI and the context

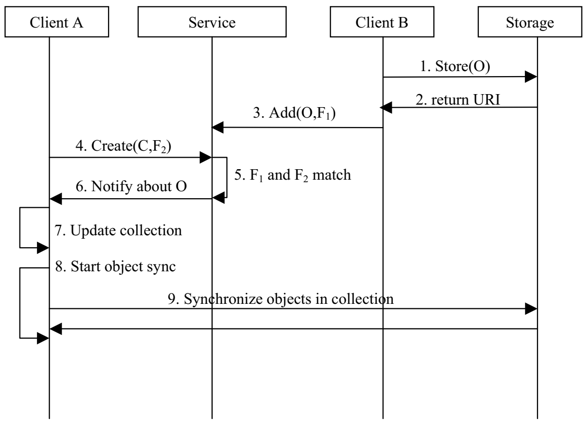


Fig. 3. Sequence diagram of synchronization without pub/sub operations

profile F_1. For example, the filter could contain information about a workshop: the duration, the location, presenters, participants, and the topics of the workshop. In 3. B adds the description with the associated filter to the synchronization service.

Client A creates a collection (4) with the context filter F_2. The filter could, for example, specify that all objects created in technical workshops where Amy, John, and Bob were present. In 5. the filters match, and A is notified about a new member in the collection (6), which is updated (7). After this the collection is synchronized using the *sync* operation (8) and the documents are updated/retrieved and possibly merged (9).

4 Example Application: Context-Aware Photo Library

To demonstrate the proposed context and metadata-based synchronization, we extended an existing synchronizing photo-library application [17] to support context-aware operation. This required three modifications: first, the application attaches metadata pertaining to the current context to each photo. The metadata is acquired from various sources, for example location information may be obtained using a GPS device or GSM cell-identifier. The second modification updates this information to the synchronization service that is reachable through the pub/sub network and allows the client to receive change notifications using either push or pull mode. The pull mode is unintrusive and does not incur any communication cost that the user is not aware of. The push mode, on the other hand, allows proactive tracking of shared photos. The third modification uses the collection API for tracking and synchronizing images.

The context-aware photo application uses the proposed mechanism as follows: 1. Context information is attached to new photos. The context information may be input from the user, gathered from various sensors, or a combination of them. In our example, we use pre-defined context descriptions. 2. The photos are published to the synchronization service using the *add* API method. This is implemented in the client application, which makes all photos in the assigned directories available to others using this API call. 3. The user instructs the application on what contexts to track and synchronize by formulating context queries. An example context query could be: $\{$ *type* = "photo" $\wedge$ location = "Helsinki" $\wedge$ description *contains* ("concert" $\vee$ "rock") $\}$. 4. The application will receive updates on any objects whose profile matches with the active queries. 5. This collection is shown to the user, and depending on the mode of operation, either all or selected, elements of this collection are synchronized. The photo library synchronizes all images.

This example illustrates how the proposed mechanism is used in context-aware applications. This same approach could be used, for example, to track documents in a collaborative environment.

5 Related Work

Novel context-aware applications have been proposed, for example Mobile Media Metadata [18], but typically many applications rely on centralized servers and do not have specific middleware support for tracking and managing objects by their context. The main goal of the proposed system is to provide this middleware support for context-aware applications.

Context-awareness is an active research topic and many systems provide support for context-aware operation to applications. Systems such as the Context Toolkit [19], Gaia [20], and the Context Cube [21] support context-aware applications. Almost all context-aware systems employ some kind of asynchronous communication abstraction, typically asynchronous events. Events support context-triggered actions, and allow run-time binding of components supporting modularity. In addition to push communication semantics many systems, such as Gaia, also support a query interface for receiving (pulling) context information.

Most systems do not have middleware support for synchronizing objects and directories based on context. Gaia has a context file system, which supports the naming and location of files based on the file path identifier. Gaia's context file system allows the attachment of context attribute to files or directories [22]. The proposed system is based on a similar idea, but the systems have several differences. First, the proposed mechanism is based on filters and uses the covering relations for matching. This allows automatic categorization of objects based on their context and the context descriptions may contain predicates. Second, the context file system is more concerned with personal data and making it available in different active spaces than distributed data synchronization and change notification. Tuple spaces, namely Lime [23], support synchronization of the tuple space, but do not offer any specific API support for continuously tracking context changes and then synchronizing a select subset of objects based on the current context.

6 Conclusions

In this paper we presented a new mechanism for context and metadata-based collection and object synchronization. We examined three important parts: the description of context and separation of the actual object data from the metadata, continuous collection synchronization, and object and directory synchronization. We focused on the first two parts and showed how the pub/sub paradigm may be used to implement collection synchronization and how filters may be used to represent both profiles and queries.

By leveraging an existing pub/sub system with support for expressive communication we considerably simplify the implementation of the system. The pub/sub paradigm supports the separation of concerns, because it decouples entities from each other and abstracts issues such as disconnections, message buffering, and mobility management. Existing algorithms for matching, covering, and overlapping allow straightforward implementation of the components. Distribution of service functionality may be realized by distributing different context types or schemas to different servers.

References

1. Dey, A.K., Abowd, G.D.: Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology (1999) ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf.
2. Kangasharju, J., Lindholm, T., Tarkoma, S.: Requirements and design for XML messaging in the mobile environment. In Anerousis, N., Kormentzas, G., eds.: Second International Workshop on Next Generation Networking Middleware. (2005) 29–36
3. Tarkoma, S.: Event dissemination service for pervasive computing. In: Advances in Pervasive Computing, Austrian Computer Society (2004)
4. Lindholm, T., Kangasharju, J., Tarkoma, S.: A hybrid approach to optimistic file system directory tree synchronization. In: Fourth International ACM Workshop on Data Engineering for Wireless and Mobile Access. (2005)
5. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Computing Surveys **35** (2003) 114–131
6. Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and evaluation of a wide-area event notification service. ACM Transactions on Computer Systems **19** (2001) 332–383
7. Mühl, G., Ulbrich, A., Herrmann, K., Weis, T.: Disseminating information to mobile clients using publish/subscribe. IEEE Internet Computing **8** (2004) 46–53
8. Loke, S.W., Padovitz, A., Zaslavsky, A.: Context-based addressing: The concept and an implementation for large-scale mobile agent systems using publish-subscribe event notification. In: DAIS 2003. (2003) 274–284
9. Carzaniga, A., Wolf, A.L.: Content-based networking: A new communication infrastructure. In: NSF Workshop on an Infrastructure for Mobile and Wireless Systems. (2001)
10. Cugola, G., Di Nitto, E., Picco, G.P.: Content-based dispatching in a mobile environment. In: Workshop su Sistemi Distribuiti: Algorithmi, Architectture e Linguaggi. (2000)
11. Keidl, M., Kreutz, A., Kemper, A., Kossmann, D.: A publish & subscribe architecture for distributed metadata management. In: Proceedings of the 18th International Conference on Data Engineering (ICDE'02). (2002)
12. Tarkoma, S., Kangasharju, J.: A data structure for content-based routing. In Hamza, M.H., ed.: Ninth IASTED International Conference on Internet and Multimedia Systems and Applications, ACTA Press (2005) 95–100
13. Carzaniga, A., Wolf, A.L.: Forwarding in a content-based network. In: Proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany (2003) 163–174
14. Mühl, G.: Large-Scale Content-Based Publish/Subscribe Systems. PhD thesis, Darmstadt University of Technology (2002)
15. Aguilera, M.K., Strom, R.E., Sturman, D.C., Astley, M., Chandra, T.D.: Matching events in a content-based subscription system. In: Symposium on Principles of Distributed Computing. (1999) 53–61
16. Fiege, L., Zeidler, A., Buchmann, A., Kilian-Kehr, R., Mühl, G.: Security aspects in publish/subscribe systems. In: Third Intl. Workshop on Distributed Event-based Systems (DEBS'04), Edinburgh, Scotland, UK (2004)
17. Lindholm, T.: XML three-way merge as a reconciliation engine for mobile data. In: Third ACM International Workshop on Data Engineering for Wireless and Mobile Access. (2003) 93–97

18. Sarvas, R., Herrarte, E., Wilhelm, A., Davis, M.: Metadata creation system for mobile images. In: MobiSys, USENIX (2004)
19. Salber, D., Dey, A.K., Abowd, G.D.: The context toolkit: Aiding the development of context-enabled applications. ACM CHI '99 (1999)
20. Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: A middleware infrastructure for active spaces. IEEE Pervasive Computing **1** (2002) 74–83
21. Harvel, L.D., Liu, L., Abowd, G.D., Lim, Y.X., Scheibe, C., Chatham, C.: Context cube: Flexible and effective manipulation of sensed context data. In: PERVASIVE 2004, Vienna, Austria, April 21-23, 2004, Proceedings. (2004) 51–68 Lecture Notes in Computer Science 3001 Springer.
22. Hess, C.K., Campbell, R.H.: A context-aware data management system for ubiquitous computing applications. In: ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems, Washington, DC, USA, IEEE Computer Society (2003)
23. Murphy, A., Picco, G., Roman, G.C.: Lime: A middleware for physical and logical mobility. In: Proceedings of the The 21st International Conference on Distributed Computing Systems, IEEE Computer Society (2001) 524–536

Facilitating Context-Awareness Through Hardware Personalization Devices: The Simplicity Device

J. Papanis, S. Kapellaki, E. Koutsoloukas, N. Dellas, G.N. Prezerakos[1], and I.S. Venieris

Intelligent Communications & Broadband Networks Laboratory,
School of Electrical and Computer Engineering, National Technical University of Athens
{jopapan, sofiak, lefterisk, ndellas, prezerak}@telecom.ntua.gr, venieris@cs.ntua.gr

Abstract. The paper presents the specification, development and initial performance evaluation of the Simplicity Device (SD), which is one of the key components of the SIMPLICITY Architecture. The SD is both a portable personalization device and a hardware Single-Sign-On (SSO) token. It accommodates the Simplicity User Profile (SUP), which contains user context related information and has been designed according to 3GPP and W3C standards. In cooperation with a distributed brokerage framework the SD provides the users with the means to automatically personalize terminals and services according to their context by the simple act of "plugging" the SD into any SIMPLICITY compliant user equipment. Within this paper we present the generic SD architecture which is the basis for different SD implementations and consequently we focus on a JavaCard SD implementation and its performance evaluation.

1 Introduction

One of the major goals of pervasive computing is to provide highly personalized context-aware services with the least possible user interference. These services usually require more complex transactions than the traditional client-server model, where the information and the required resources were conveniently stored centrally at the network side and managed by the service provider alone. Services have evolved towards a decentralized compound model; resources are distributed throughout the network but are also located in the user's environment in the form of embedded devices with networking capabilities. All these entities must cooperate towards successful service provision. User related information needs to be retrieved and processed in order for context to be formed and exploited. Consequently service provision should adapt to user context.

We present the approach followed by the IST SIMPLICITY project (Secure, Internet-able, Mobile Platforms LeadIng CItizens Towards simplicity) [1]. SIMPLICITY aims at users of existing as well as emerging services who employ different terminals (including PCs, PDAs and mobile phones) over wired and wireless networks. The user should enjoy personalized context aware services with minimal configuration

[1] G. N. Prezerakos is also with the Technological Education Institute (TEI) of Piraeus, Dpt. of Electronic Computing Systems, 250 Thivon Av. & Petrou Ralli, 122 44 Athens, Greece.

overhead. SIMPLICITY tackles complexity by combining the following means: (a) A personalization device with a simple and uniform mechanism for customizing services and devices (used for user identification / authentication as well as a means to store user profiles, preferences and policies) (b) a distributed brokerage framework that encompasses the various components that constitute an end-user application.

The personalization device proposed by the SIMPLICITY, aptly named Simplicity Device, is a portable and physically robust token that the user can carry everywhere and anytime. It helps the user interact with terminal devices and services. Users can personalize terminals and services by the simple act of “plugging” their SD into the chosen terminal. Depending on the different capabilities of the SD implementation, it can store the user profiles, user preferences, user policies and terminal settings and/or be a pointer to information stored elsewhere.

Before delving into the details of the SD, the following section gives an overview of the end-to-end SIMPLICITY architecture.

2 The Simplicity Architecture

SIMPLICITY is a distributed system that delivers advanced personalization for users, through context awareness and policy based adaptation. Overall, SIMPLICITY introduces the novel idea of a personalization device, the Simplicity Device (SD) which is supported by a distributed brokerage framework [2].

The central entity in this framework is the Simplicity Broker, a compound software entity that consists of individual subsystems delivering Simplicity services to users. Different broker setups have been specified in SIMPLICITY, most notably the Terminal Broker (TB) which resides in user terminal equipment and is responsible to deliver personalization of the terminal environment for the user, the Network Broker (NB) which resides at the network side and mediates between the Terminal Broker and network side services, and the Service Broker (SB) which hosts interfaces, APIs and adaptors for 3rd party Simplicity compliant services. The user’s interface to Simplicity is the Simplicity Personal Assistant (SPA), an entity that manages the graphical user interfaces, takes actions on behalf of the user and orchestrates the brokerage framework through the Terminal Broker. Existing hardware and software outside Simplicity, collectively labeled “legacy entities (LE) integrates with Simplicity through the use of special adaptor interfaces”.

Simplicity brokers are modular architectural entities, consisted of loosely coupled software entities called Subsystems. Subsystems are autonomous entities that encapsulate Simplicity specific functionality; they cooperate through the exchange of messages in an asynchronous event-based fashion. Subsystems are attached in a central lightweight entity, the Mediator, which is responsible for filtering messages and delivering them to their intended recipients. A special kind of subsystem, the Adaptor subsystem is the intermediate entity between legacy entities and Simplicity. The Simplicity Broker Communication (SBC) subsystem is another special subsystem that fulfils the task of broker discovery and communication with remotely attached subsystems. Finally, the Simplicity Device Access Manager (SDAM) is a special case of an adaptor subsystem that provides interfaces for different SD realizations and abstracts the different realizations to the rest of the Simplicity system.

Figure 1 illustrates the internal broker architecture and lists the aforementioned core subsystems.

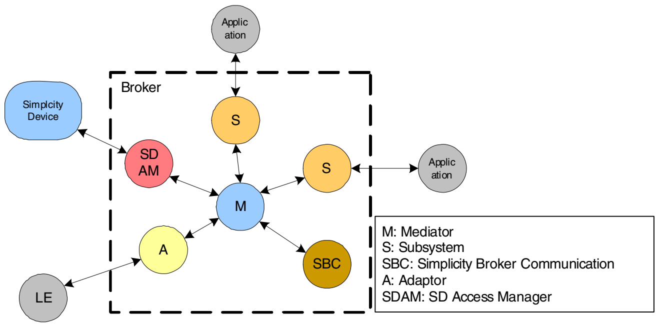


Fig. 1. Internal Broker Architecture

The focus in the following sections will be on the Simplicity Device implementation and its interaction with the Simplicity Device Access Manager.

3 Simplicity Device Functionality

Being the innovative part of the Simplicity vision, the Simplicity Device has a prominent role in the Simplicity Architecture. The SD is seen as something more than a secure storage token; it is a personalization enabler that inflicts on users' computing and networking environment a certain degree of mobility. A smart and personalized environment follows the user anywhere that a Simplicity compliant terminal can be found.

In order to realize this task, the Simplicity Device makes use of a structured user profile that contains personalization information about the user himself, about the available services, the network connectivity options, the device capabilities and the terminal environment preferences. The Simplicity User Profile (SUP) was modeled according to the methodology proposed by 3GPP for Generic User Profile (GUP) [3] and device related profiles where developed based on the W3C Composite Capabilities/Preferences Profile framework (CC/PP) [4]. SUP is stored on a Simplicity Device realization and the device implements store, retrieve and verify procedures while operating on the SUP, thus ensuring data integrity as well as placing access on SUP under the device's strict security control. Depending on the different capabilities of the device implementation, it can store the user profiles, user preferences, user policies and terminal settings locally and/or be a pointer to information stored at specific nodes in the network. Additional information about the relation between SUP, the GUP and CC/PP can be found at [5].

Besides the personalization features, the SD provides single-sign-on (SSO) functionality to the user. Seamless service provision requires that users entering a service provisioning framework are identified only once and consequently they can use services based on their access rights and personal preferences. Several SSO solutions have been proposed [6], [7]. The most publicized frameworks are the Microsoft's Passport [8] and the Liberty Alliance Project [9], both proposing that the user's profile is stored at a central point within the network. The innovative feature, however, that Simplicity projects, is the use of a hardware SSO token, the SD, which is both more secure and more practical to use, since the steps required are to plug in the device and provide a PIN number to the Simplicity Personal Assistant.

Personalization and the single-sign-on function are the two key features that the Simplicity Device contributes to the Simplicity vision. The following section investigates the requirements that arise from these features and describes the architecture and implementation of a JavaCard based Simplicity Device.

4 The Simplicity Device Architecture and Implementation

Summarizing the facts from the previous section, an ideal SD should meet the following requirements:

- Portability (small size, lightweight, low power consumption)
- Plug & play & unplug (bootstrap procedure, insert and removal notification)
- Computational Power (profile management, security algorithms)
- Data Storage
- Security (Authentication\Encryption)
- Ease of programming

Unfortunately some of the above requirements are conflicting. But if we had to choose one of them, the most important is portability with an emphasis on small-sized devices, which are resource-constrained. Thus the personalization device should be offered in different implementations that integrate with currently available and emerging terminals with minimum effort. Three main SD types can be defined taking also into account the fact that the functionality of the SD has to be implemented on resource constrained devices:

- Storage and processing devices offering a rich set of functions and facilities, exploiting the storage and processing capabilities of the device.
- Pure storage devices with a large storage capability allowing the storage of significant volumes of data
- Virtual devices consist of a software solution based on a network infra structure

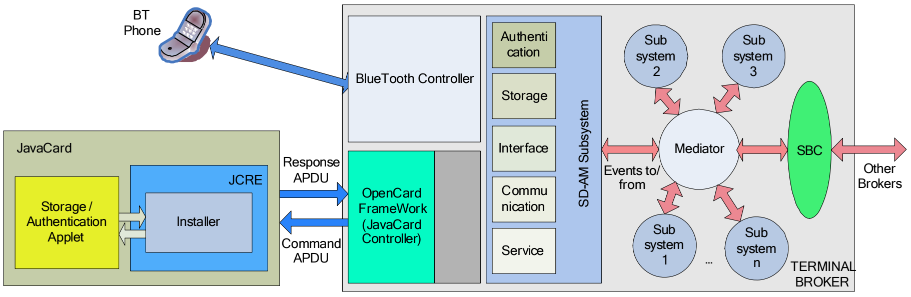


Fig. 2. Simplicity Device Architecture

Given the above options the SIMPLICITY project chose to implement a JavaCard [10] based SD and a legacy SIM card combined with Bluetooth [11] mobile phone based SD. Both of these implementations are portable, can easily connect with a wide selection of terminals (JavaCard uses the USB interface and the SIM realization uses the wide spread Bluetooth protocol) and also offer some processing power and storage space. In order to overcome the capacity limitations posed by the SD hardware, a part of the SD software entities is located in the terminal and a part in the physical SD. Figure 2 contains the main blocks of the SD and terminal architecture.

In the following section the paper focuses mainly on the JavaCard / USB SD implementation which was specified and developed by our group.

4.1 SD Terminal Part

Every terminal subsystem should be able to cooperate with any SD type. For this reason a special subsystem called the SD Access Manager (SDAM) was created which:

- provides uniform interface for all classes of SD implementations towards the terminal broker
- is responsible to provide access to functionality originally assigned to the personalization device, even if this functionality is provided by some other component, due to restrictions of the specific personalization device implementation.

SDAM's API supports both types of personalization devices (JavaCard & Bluetooth phone). The terminal broker subsystems should not bear the burden of distinguishing to which type a specific SD implementation belongs to. Any functionality that is not directly supported by a specific SD implementation is provided by the SDAM. In order to achieve this, the SDAM consists of a common part and specific entities, called Controllers, for each implementation. The common part is split into five managers, which are described in the following paragraphs.

The Authentication Manager is responsible for authentication and security issues regarding the SD. The tasks include authentication of the user towards the SD (performed by entering a Personal Identifier Number (PIN) code) and authentication of the SD towards the network (in order to access the Simplicity services). Additionally specific 3rd party services and applications that may try to access and possibly modify data stored in the SD should have the proper access rights. The Storage Manager component is responsible for the management of the data stored in the physical SD. It allows accessing the contents of the SD by implementing a basic file-system, providing file-system operations like read, write, delete and update. The Interface Manager is responsible for handling the Simplicity events coming from the Mediator, forwarding them to the corresponding components and also encapsulating the responses from the components into Simplicity events that are transmitted back to the Mediator. For example a request such as "get user's billing preferences" will be interpreted into downloading a specific part from the SUP. The Interface Manager is also involved in the bootstrapping procedure of the SD. The Communication Manager component is another vital component for the proper function of the SDAM. The most important services are presence detection of a connected SD, handling of status changes (insertion and removal), discovering of SD capabilities and finally instantiation of the respective Controller. Another responsibility is the discovery of any available online data storage that may hold SD data and then acting as a proxy towards it. Finally, the Service Manager component is offering a collection of services to other components, which focus on improving the overall speed - performance and security of the subsystem. The available services include caching profile data from the physical SD, encoding, decoding and encryption of data and a finally maintaining a permission access table.

The JavaCard Controller is a high level interface, which resides in the terminal part and is responsible for hiding the complexity of the JavaCard implementations from the SDAM. Therefore terminal software developers are not concerned with the internal organization of the JavaCard and its associated concepts such as Application Protocol Data Units (APDUs), JavaCard Applets, Application Identifiers (AID) etc. In order to achieve this we followed the OpenCard Framework (OCF) [12], which communicates via appropriate drivers with the programs (applets) running in the JavaCard SD through the USB port. By using the OCF, we avoid making vendor specific software, i.e. the terminal side becomes rather versatile and should work with smart cards from numerous vendors. When the JavaCard Controller receives a request from the SDAM, it forms a suitable APDU and passes it to the OCF that forwards it to the physical JavaCard via USB. When it receives the response from the JavaCard, it decomposes the APDU(s) and forwards the valuable information to the SDAM. The complexity of the process is alleviated by the Controller, for example when writing a file to the JavaCard, the current implementation of the Controller has to split the file in segments that fit in the APDU's payload size and also maintain and update a mapping between files (user profile, preferences) and storage positions in the memory of the card.

4.2 SD Physical Part

Multifunctional smart cards based on JavaCard technology can be implemented in two different ways. The first and probably most obvious is to create a separate applet for each command. This method relies a lot on the card issuers to implement their cards correctly and with a decent security scheme where necessary. A major drawback is, that the various applets know nothing of each other, so there's a strong need for the applets to implement certain interfaces, in order to perform inter-applet communication. The inter-applet communication should be considered a necessity, as the applets will share objects like PIN and SUP. Furthermore, the procedure of uploading and installing an applet onto a card is proprietary and tedious. Another strategy which is the one used in our implementation is to implement only a single applet and let that applet control everything. The applet handles creation and modification of the data and can be reached through OCF applications, which makes it easy to deliver a well-defined interface to any developer.

The physical mapping on a JavaCard foresees the use of memory locations reserved inside the context of a JavaCard applet. These memory locations are allocated by means of a memory buffer constituted by one or more byte array (depending on the memory constraints of the card). The communication between a device equipped with a smart card reader (PC, PDA, Phone, etc.) and the JavaCard is based on the APDU protocol mapped into interface primitives. ISO 7816-4 [13] defines the APDU protocol as an application-level protocol between a smart card and an application on a suited device. The first APDU is addressed to the installer applet of the JavaCard, which is provided by the vendor by default. This command selects the suitable applet for the operation we need to perform. After selecting the applet, all following APDUs are directed to that applet and processed by its process method, until a deselect command is issued or another applet is selected.

A storage/authorization applet realizes a portable data memory with standard ISO/IEC 7816-4 commands. In order to read, write or delete data, the user has to authenticate using the PIN verification, which is stored in the card. Inside the data objects are organized in ASN.1 BER TLV coded data objects. Every data object has its own tag and length, which are used in the APDUs for read and write. The applet was developed on the GemXplore Developer environment which includes an Integrated Development Environment bundled with JavaCard 2.1 compliant smart-cards [14]. Figure 3 displays the code that is executed by the Controller during the authentication.

```
public boolean activateSD(String pass)
  throws AuthenticationException {

  String[] passp = pass.split(" ");
  byte[] Pinn = new byte[passp.length];
  for(int i=0;i<passp.length;i++)
    Pinn[i] = Byte.parseByte(passp[i]);
  ResponseAPDU myResponse = null;
  byte bZero = (byte)0x00;
  byte verify_position = (byte)0x01;
  CommandAPDU request =
    new CommandAPDU(5+Pinn.length);
  request.setLength(0);
  request.append(CLASS);
```

```
  request.append(VERIFY_INS);
  request.append(bZero);
  request.append(verify_position);
  request.append((byte)Pinn.length);
  for(int i=0; i<Pinn.length; i++)
    request.append(Pinn[i]);
  card.beginMutex();
  myResponse =
    sdjCardProxy.send(request);
  if (myResponse.sw() ==  0x9000)
    return true;
  else return false;
  card.endMutex();
}
```

Fig. 3. Authentication method at the Controller

First, a string password is converted into bytes (JavaCard has no String) and a proper request Apdu is formed by appending APDU CLASS, INS, lengths and payload (password). Consequently, the card is being contacted, the Apdu is sent and the response is read (0x9000 is the OK code).

5 Performance Evaluation of the JavaCard SD

The objective of our measurements is to obtain information concerning the time needed to read and write data from and to the memory of the JavaCard. The measurements concern the reading and writing of a number of raw data with length varying from 1 to 10000 Bytes. More specifically we obtained 10 measurements of 1, 15, 50, 100, 200, 300, 500, 1000, 5000 and 10000 Bytes with a minimum of 100 samples per measurement. In order to measure the times we have used the standard method

System.currentTimeMillis() supplied from the Java Core (we used is JRE 1.4.2_01-b05 for our measurements). This method returns the current time in milliseconds, which is precise enough for our measurements.

The input Bytes were read from files with appropriate sizes (1 Byte to 10KBytes – the size of the files was actual Byte count, not the size these files occupy on disk). We measured the time needed to format the APDUs (set memory position to read or write data and apply data in case of write), send them and receive response from the card (data included also in case of read).

The JavaCard that we used for our implementation and measurements is the GemXplore Xpresso v3.2 from GemPlus, which comes together with a USB Card Reader. The PC which was connected to the card was an Intel Pentium III 1.8 MHz with 512 MBs of RAM. Finally the USB port of the PC is version 1.1. The measured mean times for varying-length bulk data read and write operations are presented in Table 1.

Table 1. JavaCard-based SD Measurements

Bytes	Read time (ms)	Write time (ms)
1	20	33
15	20	33
50	20	33
100	20	33
200	40	66
300	60	100
500	82	134
1000	167	273
5000	1010	1542
10000	2465	3523

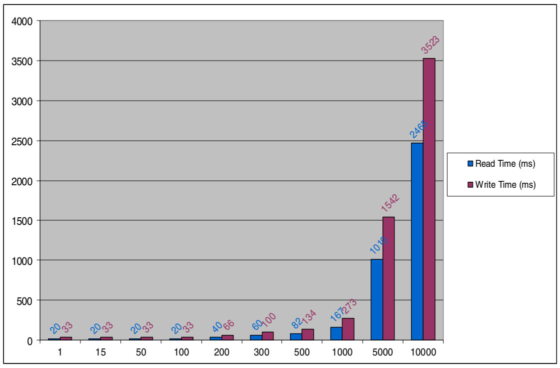


Fig. 4. Read and Write times in ms

Figure 4 shows a graphic representation of the measurements. One can notice that the time needed for reading is always somewhat less than the time needed for writing, which is something we expected because data are read from / written to EEPROM memory and the writing EEPROM operations are slower that the reading ones.

Our specific implementation is another reason, since everyone can read from the card but in order to write the applet checks if the user is authorized.

In figures 5 and 6 we present results for different numbers of APDUs. Figure 6 shows a detail of Figure 5 from the first APDU till the eighth. One interesting observation is that for few APDUs (in our case the first 8) the measured times are multiples of the time required for one APDU. For example, if one APDU requires 20ms for reading and 33ms for writing, 8 APDUs require 167ms for reading, which is almost equal to 8 x 20 (160ms) and 273ms for writing, which is almost equal to 8 x 33 (264ms). For larger numbers of APDU this linear relationship no longer applies. An average SUP will be around 5 to 10 KBytes long. Our measurements suggest that writing/reading an average SUP to/from the JavaCard SD will require just a few seconds (2.5 for reading and 3.5 at most for writing). Reading/writing specific data (e.g. a username – 50 Bytes will just take 20/33 ms).

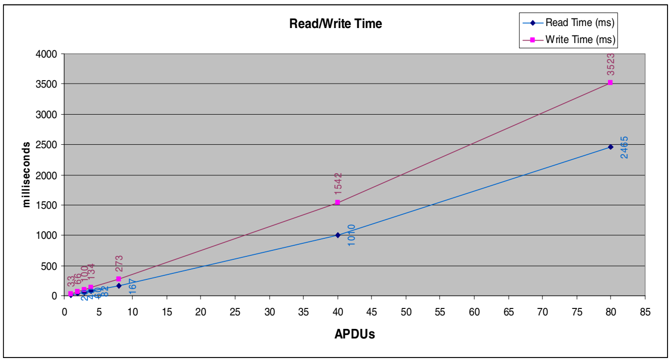


Fig. 5. Read and Write times/APDUs

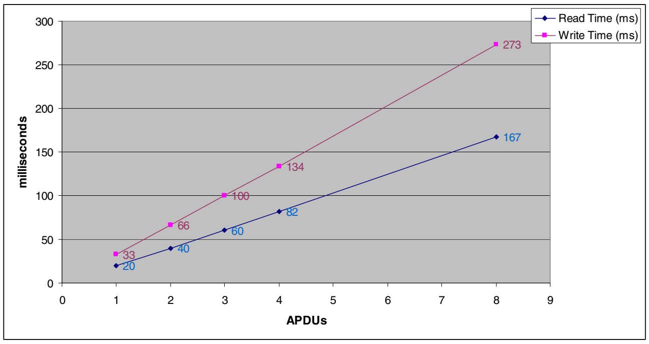


Fig. 6. Read and Write for 1 to 8 APDUs

In the future we intend to repeat these measurements on the final version of the JavaCard SD, which will provide optimized exchange of messages and maybe even a compression option for data stored in the memory of the SD. Since the data is in text format the compression will reduce the size significantly. Thus, we expect that the final version will offer significant performance enhancements.

6 Conclusions and Future Work

The design, implementation and initial performance evaluation of a JavaCard based SD, in the framework of the SIMPLICITY architecture, has been extensively described within this paper. This personalization device with the usage of the SUP and the support of a distributed brokerage framework provides the users with the means to enjoy simplified, personalized and automated procedures for the usage of different terminals and services.

Ongoing work in SIMPLICITY aims at the integration of a number of applications to a fully working demonstrator. Depending on the application, a variety of terminals (laptops, PDAs, mobile phones) will be used via wireline and wireless connections combined with the different SD implementations. While the demonstrator will provide proof-of-concept of the SIMPLICITY architecture and the SD concept, on the other hand it is not a sufficient means for thoroughly evaluating the performance of the overall SIMPLICITY System as well as the SD part. For this reason, we are currently working on appropriate performance models (analytical and simulation-based) that will produce more detailed results and give a more pragmatic insight of the system's real operation. Also comparisons of the different SD implementations are included in future steps.

Acknowledgement

The ideas that are described in this paper originate from the ongoing work in the Simplicity project, but most of them represent the personal elaboration of the authors. The authors wish to express their gratitude to the people working in the Simplicity Consortium.

References

1. SIMPLICITY Project, http://www.ist-simplicity.org
2. N. Blefari Melazzi, S. Salsano, G. Bartolomeo, F. Martire, E.Fischer, C. Meyer, C. Niedermeier, R. Seidl, E. Rukzio, E. Koutsoloukas, J. Papanis, I. S. Venieris: "The Simplicity System Architecture", to appear in the Proceedings of the 14th IST Summit, 19-23 June 2005, Dresden, Germany.
3. GPP TS 23.240, "3GPP Generic User Profile (GUP) requirements; Architecture (Stage 2)"
4. W3C Recommendation 15 January 2004, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0"
5. E. Rukzio, G. N. Prezerakos, G. Cortese, E. Koutsoloukas, S. Kapellaki. "Context for Simplicity: A Basis for Context-aware Systems Based on the 3GPP Generic User Profile", International Conference on Computational Intelligence (ICCI 2004), pp. 466-469 ,17th-19th December 2004, Istanbul, Turkey
6. J. Futagawa, "Integrating Network Services of Windows and UNIX for Single Sign-On", Proceedings of the 2004 International Conference on Cyberworlds (CW'04), pp. 323-328, November 2004, Tokyo, Japan

7. G. Zhao, D. Zheng, K. Chen,“ Design of Single Sign-On“, Proceedings of the E-Commerce Technology for Dynamic E-Business, IEEE International Conference on (CEC-East'04), pp. 253-256, September 2004, Beijing, China
8. Microsoft Corporation, “.NET Passport: Balanced Authentication Solutions”. http://www.microsoft.com/net/services/passport/balanced.asp
9. Liberty Alliance Project, http://www.projectliberty.org/
10. Java Card technology specification available at http://java.sun.com/products/javacard/index.jsp
11. Bluetooth Official site, https://www.bluetooth.org/
12. OpenCard Framework web site, http://www.opencard.org/
13. ISO/IEC 7816-4:1995 Command set for microprocessor cards.
14. GemXplore Developer, http://www.gemplus.com/products/gemxplore_developer/

Network Access Security Management (NASM) Model for Next Generation Mobile Telecommunication Networks

Muhammad Sher and Thomas Magedanz

Technical University Berlin/Fokus Fraunhofer Berlin, Germany
{sher, magedanz}@fokus.fraunhofer.de
http://www.av.cs.tu-berlin.de/
http://www.fokus.fraunhofer.de/ngni

Abstract. In this paper the main focus is to establish a set of security features, methodologies and mechanisms to build the Network Access Security Management (NASM) architecture which provides users a secure access to next generation mobile telecommunication network services. The NASM architecture involves the authentication and key agreement, security mode setup during connection establishment, and access link data integrity and confidentiality between mobile user and universal mobile telecommunication systems. The NASM framework also provides measures to enhance the security weaknesses of GSM security framework and protect the users against attacks on the radio access link of Universal Mobile Telecommunication Systems (UMTS) based on 3GPP technical specifications [1]. The NASM model is part of Secure Service Provisioning Framework (SSPF) [2] for IP Multimedia System (IMS) and Next Generation Networks (NGN) of Third Generation and beyond (3Gb) Testbed [9] and IP Multimedia System (IMS) Playground [10] which provide R& D infrastructure at Fokus Fraunhofer.

1 Introduction

The main concerns of wireless networking and mobile communication are security among service providers and end users. We need additional security measures at radio interfaces and to access wireless and mobile services to counter the interception of data on the radio interfaces and illegitimate access to mobile services. The interception of user's data causes the loss of confidentiality of sensitive user's information and the illegitimate use of services cause masquerading and false billing.

First time the security measures were taken into account in second generation digital cellular systems e.g. in GSM networks [7] by user authentication and user data confidentiality. These networks use challenge-response mechanisms to authenticate the user's identity to prevent the fraudulent use of mobile services. The main security flaw in the GSM networks is that the user cannot authenticate the serving network and causes the false base station attack and the other is indefinitely reused of triplet which is the authentication vector of GSM [8].

The Third Generation Partnership Project (3GPP) has adopted an enhanced security framework for the Universal Mobile Telecommunication Systems (UMTS) to overcome the weaknesses of security in GSM and other 2G networks. In reference of

3G security; [4], [5], [6], [1] discuss the security principles and objective, security threats and requirements, access security for IP-based services, security architecture respectively.

We have proposed a Secure Service Provisioning Framework (SSPF) [2] for IP Multimedia System (IMS) and Next Generation Mobile Network as a component of 3Gb Testbed [9] and IMS Playground [10] which provide infrastructure for next generation networks at Fokus, Fraunhofer. The paper will focus to build the Network Access Security Management (NASM) model, according to the 3GPP specification, which will provide secure access of radio link through authentication and key agreement procedure, access link data integrity and data confidentiality between users and UMTS networks as a part of SSPF [2] within IMS playground. In section 2, we will discuss the NASM conceptual model; section 3 and 4 will explain the features and mechanisms of NASM respectively and last section will conclude the paper.

2 Conceptual Model of NASM

The network access security management architecture consists of User Service Identity Module (USIM), Mobile Equipment (ME), Access Network (AN), Service Network (SN) and Home Environment (HE) as shown in *figure 1*. The USIM is required for accessing the Packet Switched (PS) domain in General Packet Radio System (GPRS) and identifies the particular subscriber. The USIM contains the security parameters for accessing the PS-domain, International Mobile Subscriber Identity (IMSI), list of allowed access points, MMS-related information. In serving network, the Serving GPRS Support Node (SGSN) links the Radio Access Network (RAN) to the packet core network in the PS-service domain. It is responsible for performing both control and traffic handling functions for the PS domain. The control parts deal with mobility management and session management. The SGSN also ensures appropriate QoS and generates charging information [11]. In the CS-service domain, the related part is Visitor Location Register (VLR). In HE, the Authentication Centre (AUC) generates the AV vector as given in equation 1.

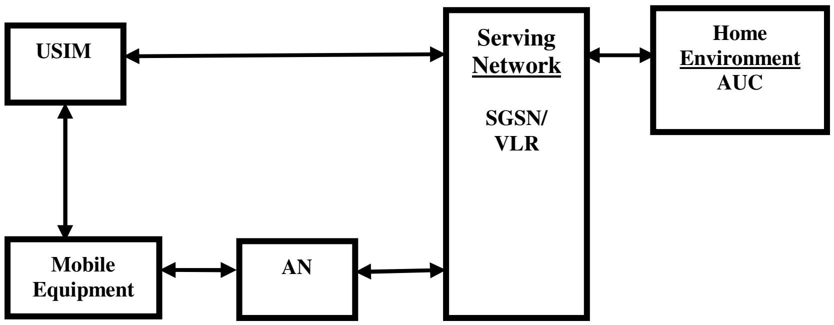


Fig. 1. Overview of NASM Model

3 Security Characteristics of NASM Model

The NASM model consists of the following four key characteristics to ensure the user security on radio access link [1]:

User Confidentiality: This entity provides user identity confidentiality, user location confidentiality and user un-traceability. To achieve these characteristics the user is assigned by a temporary identity so that the user's permanent identity to which services are delivered cannot be eavesdropped on the radio access link and in addition user data and signaling that might reveal the user's identity is ciphered on the radio access link.

Entity Authentication: Authentication entity is based on user authentication and network authentication, and should apply at connection setup between the user and the network. It involves authentication mechanism using an authentication vector delivered by the user's HE to the serving network and a local authentication mechanism using the integrity key establishment between the user and the serving network.

Data Confidentiality: It provides confidentiality of user data and signaling data. It is achieved by using cipher algorithm and key agreement.

Data Integrity: It provides data integrity and origin authentication of signaling data. Data integrity is achieved by integrity algorithms and integrity key agreement.

4 Security Methodologies and Mechanisms of NASM Model

There are five security methodologies and mechanisms to implement the architecture of NASM model to protect the user and user's sensitive information on Radio Access Network (RAN) as discussed below.

4.1 Assigning Temporary and Permanent Identities to User

In order to hide the identity of user on the radio access link, the user is assigned Temporary Mobile Subscriber Identity (TMSI) to identify the user on the radio link, for instance in paging, location update, attach, service, connection re-establishment and detach requests. The temporary identity has importance only in the location area or routing area in which the user is registered. Outside that area it should be accompanied by Location Area Identification (LAI) or Routing Area Identification (RAI) in order to avoid ambiguities. The association between the permanent and temporary user identities is kept by the Visited Location Register (VLR) for CS domain and by SGSN for PS domain depending in which domain the user is registered.

When the user could not be identified by IMSI, the serving network identifies the user by the permanent identity which is called the International Mobile Subscriber Identity (IMSI). In particular, it should be used when the user registers for the first time in a serving network, or when the serving network cannot retrieve the IMSI from the TMSI by which the user identifies itself on the radio path. The procedure is initiated by the visited SGSN/VLR that requests the user to send its permanent identity as shown in *figure 2*. The user's response contains the IMSI in clear text. This represents a breach in the provision of user identity confidentiality and in section 4.5 we will discuss the protection of user's temporary identities.

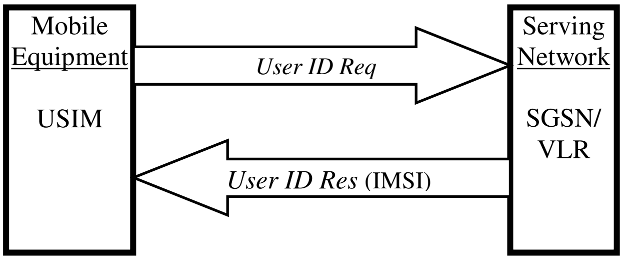


Fig. 2. Identification by the permanent identity

4.2 Authentication and Key Agreement Mechanism

The authentication and key agreement procedure involves Authentication Centre (AUC) within HE, SGSN/VLR (Visitor Location Register) and Mobile Station (MS) networks entities. The SGSN invokes the procedures by requesting authentication vectors to the AUC. The AUC generates and sends an ordered array of *m* authentication vectors to the SGSN on its requests. *AV(k)* authentication vector is used for *kth* authentication and key agreement between the SGSN and the USIM. The SGSN selects the *AV(k)* authentication vector from the ordered array and sends the parameters *RAND* and *AUTN* to the user. Particular node uses *AVs* on a first-in/first-out basis. The USIM checks whether *AUTN* can be accepted and, if so, produces a response *RES* which is sent back to the SGSN. The USIM also computes *CK* and *IK*.

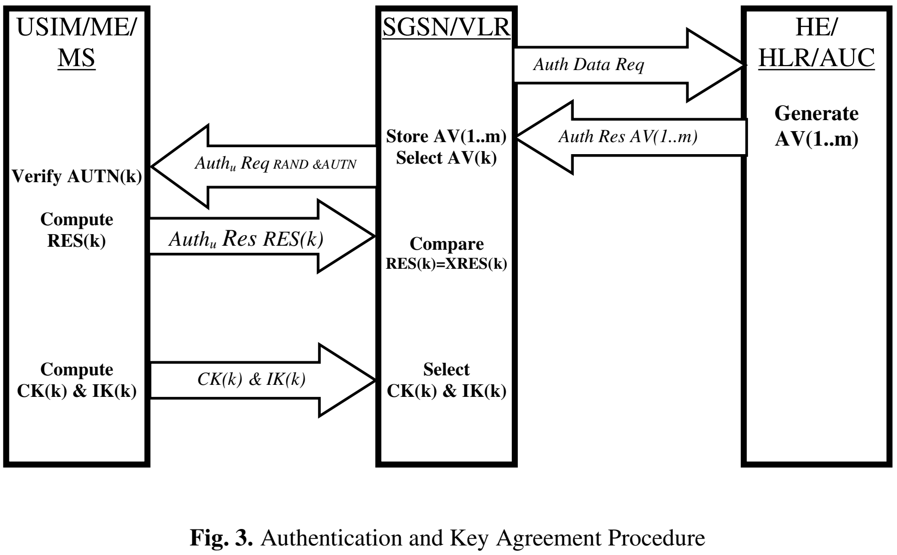


Fig. 3. Authentication and Key Agreement Procedure

The SGSN compares the received *RES* with *XRES*. If they match the SGSN considers the authentication and key agreement exchange to be successfully completed. The established keys *CK* and *IK* will then be transferred by the USIM to the SGSN for performing ciphering and integrity functions. This procedure is shown in *figure 3*. Each AV (called UMTS quintet) consists of five parameters: RANDom number *(RAND)*, eXpected RESponse *(XRES)*, Cipher Key *(CK),* Integrity Key *(IK)* and AUTHentication tokeN *(AUTN)* as given in equation 1.

$$\mathrm{AV} = \mathrm{RAND} \,\|\, \mathrm{XRES} \,\|\, \mathrm{CK} \,\|\, \mathrm{IK} \,\|\, \mathrm{AUTN}. \tag{1}$$

The above five entities of AV vector are calculated according to equations 2 to 6 and procedure is given in *figure 4*. The AUC first generates sequence number *(SQN)* and random challenge *(RAND)* and uses secret key *(K)* which is shared between USIM and AUC, and Authentication & Key Management Field *(AMF)* which defines operator specific options in the authentication process, for example, the use of multiple authentication algorithms or a limitation of key lifetime.

$$\mathrm{CK} = \mathrm{f3}_{\mathrm{K}}(\mathrm{RAND}). \tag{2}$$

$$\mathrm{IK} = \mathrm{f4}_{\mathrm{K}}(\mathrm{RAND}). \tag{3}$$

$$\mathrm{MAC} = \mathrm{f1}_{\mathrm{K}}(\mathrm{SQN} \,\|\, \mathrm{RAND} \,\|\, \mathrm{AMF}). \tag{4}$$

$$\mathrm{XRES} = \mathrm{f2}_{\mathrm{K}}(\mathrm{RAND}). \tag{5}$$

$$\mathrm{AK} = \mathrm{f5}_{\mathrm{K}}(\mathrm{RAND}). \tag{6}$$

Where, f1 = the network authentication function; f2 = the user authentication function; f3 = the cipher key derivation function; f4 = the integrity key derivation function; f5 = the anonymity key derivation function; $\|$ *denotes the concatenation.*

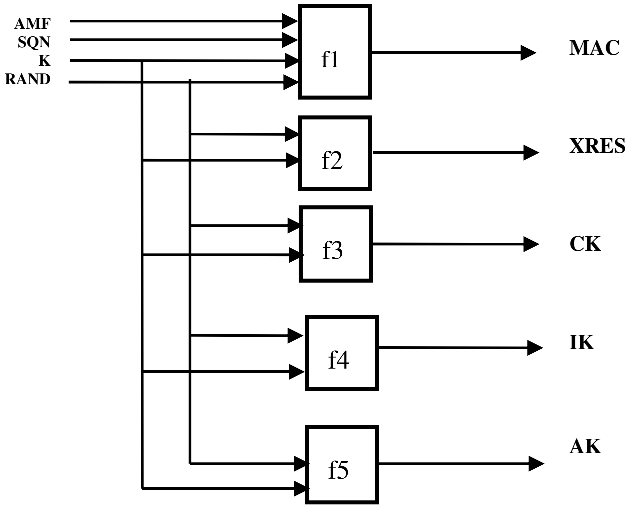


Fig. 4. Authentication Vector Generation by AUC

The detail description and explanation of these algorithms can be found in [12]. The SGSN calculates the *AUTN* entity according to equation 7 and sends this value to user in the user's request along with *RAND*.

$$\text{AUTN} = \text{SQN} \oplus \text{AK} \parallel \text{AMF} \parallel \text{MAC} \tag{7}$$

After receiving the authentication request from SGSN which includes RAND and AUTN, the USIM first computes the anonymity key *(AK)* as given in equation 5 and retrieves the sequence number *(SQN)* using equation 8. The AK is used to conceal the sequence number as the latter may expose the location of the user to protect against passive attacks only. If no concealment is required, AK = 0 (f5 = 0).

$$\text{SQN} = (\text{SQN} \oplus \text{AK}) \oplus \text{AK}. \tag{8}$$

Then the USIM computes the *XMAC* according to equation 9 and compare this with *MAC* included in *AUTN*. If they are different, the USIM sends user authentication reject message to the SGSN and abandons the procedure. The user authentication function in the USIM is shown in *figure 5*.

$$\text{XMAC} = \text{f1}_K\ (\text{SQN} \parallel \text{RAND} \parallel \text{AMF}). \tag{9}$$

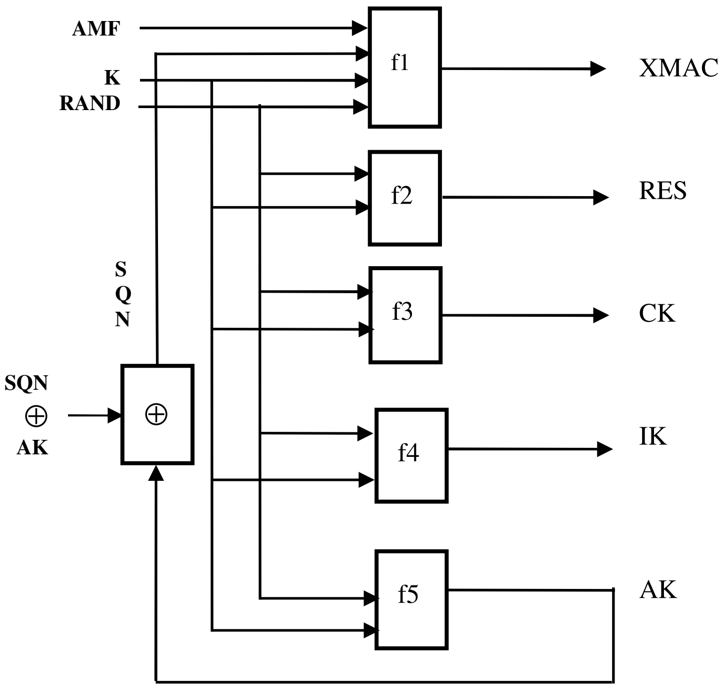


Fig. 5. User Authentication Function in USIM

The HE and USIM keep track of counters SQN_{HE} and SQN_{ME} respectively to support network authentication. The USIM verify the *SQN* in correct range ($SQN > SQN_{ME}$), if it is not in correct range, the MS sends synchronization failure to the SGSN and abandons the procedure. In this case HE may need to resynchronize the counter SQN_{HE}. If the sequence number SQN is considered to be correct range, the authentication of the network is successful [3]. In this case the, MS computes the RES as equation 10 and send it to SGSN. Next the MS sets SQN_{ME} equal to SQN if SQN_{ME} < SQN.

$$RES = f2_K\ (RAND). \tag{10}$$

Upon receipt of the user authentication response, SN compare RES with the expected response XRES from the selected authentication vector. If RES = XRES, the authentication of the user is successful and the serving network selects the CK and IK from the selected authentication vector. If RES and XRES are different, SGSN sends an authentication failure report to HE and abandon the procedure.

4.3 Security Mode Setup During Connection Establishment

The security mode setup is achieved by the integrity protection functionality. When the SGSN is known the identity of mobile subscriber, the cipher and integrity keys setting could occurred. These keys are stored in the VLR and transferred to Radio Network Controller (RNC) as required. These keys are also stored on USIM and updated from CS domain or PS domain independently. During ME registration and connection establishment within UMTS with a CS service domain and a PS service domain, user identification, authentication and key agreement will take place independently in each service domain. The connection establishment includes the ME security capability i.e. the ciphering association and the integrity association of the MS.

4.4 Access Link Data Integrity

Most of the control signaling elements like RRC, MM, CC, GMM and SM messages between the mobile station and the RNC are considered sensitive and must be integrity protected by the message authentication function. The access link data integrity procedure is given in *figures 6a & 6b*. The input parameters to the algorithm are *IK*, integrity sequence number *(COUNT-J)*, the signaling data *(S-Data)*, the direction *(DIRECT)* and the random value generated by the network side *(R-Value)*. From these four parameters the user equipment or RNC computes message authentication code *MAC-J* for data integrity using the integrity algorithm ***f9*** as given in equation 11. The use of Kasumi for the integrity protection function ***f9*** is specified in [13]. The *MAC-J* is then appended to the message when sent over the radio access link. The receiver computes *XMAC-J* as given in equation 12, on the message received in the same way as the sender computed *MAC-J* on the message sent and verifies the data integrity of the message by comparing it to the received *MAC-J*. There may be one IK for CS connections (IK_{CS}), established between the CS service domain and the user and one IK for PS connections (IK_{PS}) established between the PS service domain and the user.

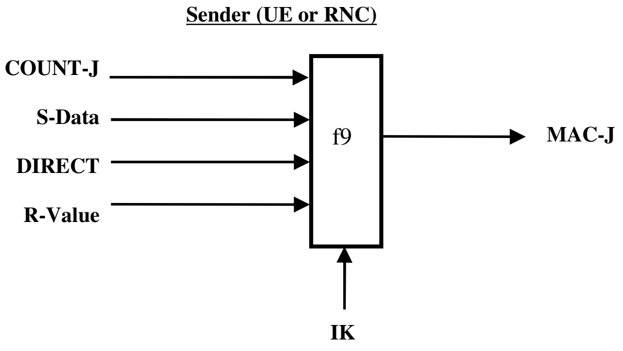


Fig. 6a. Sender's Message Authentication Function

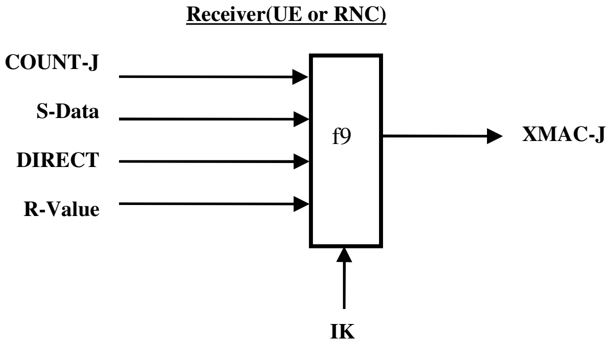


Fig. 6b. Receiver's Message Authentication Function

$$\text{MAC-J} = f9_{IK}\ (\text{COUNT-J, S-Data, DIRECT, R-Value}). \tag{11}$$

$$\text{XMAC-J} = f9_{IK}\ (\text{COUNT-J, S-Data, DIRECT, R-Value}). \tag{12}$$

4.5 Access Link Data Confidentiality

User data and some signaling elements between user and network are considered sensitive and should be confidentiality protected. To ensure identity confidentiality, the temporary user identity (TMSI) should be transferred in a protected mode at allocation time and at other times when the signaling procedures permit it. This protected mode of transmission is applied on dedicated channels between the ME and the RNC by a confidentiality function. *Figures 7a & 7b* illustrate that the sender produces keystream (Kst) by using the Ciphering Algorithm ***f8*** [13] and gets the Ciphertext (C(x)), from bit by bit binary addition of the Plaintext (P(x)) and the keystream as given in equation 13. At the receiver, the P(X) may be produced by generating the same keystream using the same input parameters and bit by bit binary addition of the C(X) and keystream as given in equation 14. The input parameters to the algorithm are the

Cipher Key *(CK)*, a time counter input *(COUNT-T)*, the bearer identity *(B-ID)*, the direction of transmission *(DIRECT)* and the length of the keystream required *(LEN)*.

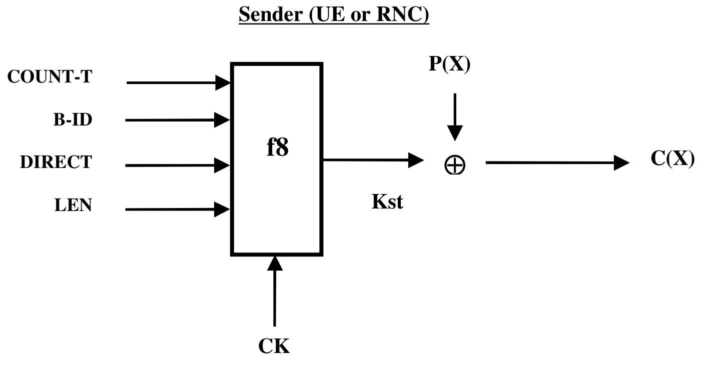


Fig. 7a. Ciphertext Production by Sender

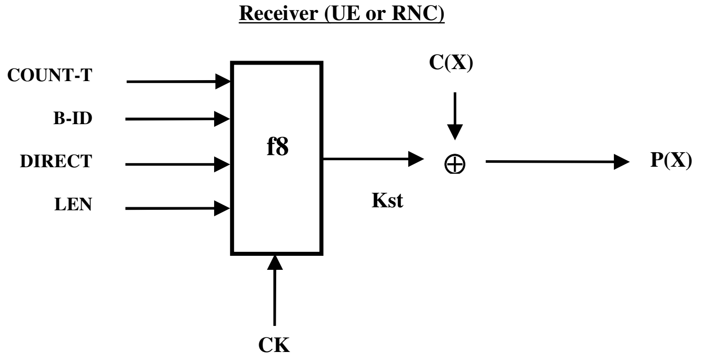


Fig. 7b. Plaintext Retrieval by Receiver

$$C(X) = P(X) \oplus f8_{CK}\,(\text{COUNT-T, B-ID, DIRECT, LEN}). \tag{13}$$

$$P(X) = C(X) \oplus f8_{CK}\,(\text{COUNT-T, B-ID, DIRECT, LEN}). \tag{14}$$

5 Conclusions

This paper provides an architectural and implementation perspective of Network Access Security Management (NASM) model for next generation mobile and wireless telecommunication systems. The proposed NASM model is based on 3GPP specifications to access the Universal Mobile Telecommunication Systems (UMTS), which is an emerging standard for 3G mobile communications and it will be a component to provide Secure Service Provisioning Framework (SSPF) environment to 3Gb.

Testbed and IMS Playground at Fokus Fraunhofer which provides research and development infrastructure for validating and implementing IP Multimedia System (IMS) and Next Generation Networks (NGNs).

References

[1] 3GPP, TS 33.102 V6.2.0 (2004-09), 3G Security; "Security Architecture (Release 6)". *www.3gpp.org*.

[2] Muhammad Sher, Thomas Magedanz, "Secure Service Provisioning Framework (SSPF) for IP Multimedia System and Next Generation Mobile Networks" 3rd International Workshop in Wireless Security Technologies, London, U.K. (April 2005), IWWST'05 Proceeding (101-106), ISSN 1746-904X. *www.iwwst.org.uk.*

[3] Muxiang Zhang, Yuguang Fang, "Security Analysis and Enhancements of 3GPP Authentication and Key Agreement Protocol" IEEE Transactions on Wireless Communications, Vol.4. No. 2 (March 2005) 734 – 742.

[4] 3GPP, TS 33.120 V4.0.0 (2001-03), 3G Security; "Security Principles and Objective (Release 4)". *www.3gpp.org*.

[5] 3GPP, TS 21.133 V4.1.0 (2001-12), 3G Security; "Security Threats and Requirements (Release 4)". *www.3gpp.org*.

[6] 3GPP, TS 33.203 V6.4.0 (2004-09), 3G Security; "Access Security for IP-based services (Release 6)". *www.3gpp.org*.

[7] European Telecommunication Standards Institute (ETSI), GSM 02.09, "Security Aspects, June 1993". http://www.etsi.org.

[8] C.H. Lee, M.S. Hwang, W.P. Yang, " Enhanced privacy and authentication for the global system for mobile communications", Wireless Networks, Vol.5 (1999) 231-243.

[9] www.fokus.fraunhofer.de/national_host.

[10] www.fokus.fraunhofer.de/ims.

[11] Poikselkae, M., Mayer, G., Khartabil, H., Niemi, A., 2004. "The IMS, IP Multimedia Concepts and Services in the Mobile Domain", John Willey & Sons Ltd, West Sussex, England (2004).

[12] 3GPP, TS 35.205 V6.0.0 (2004-12), 3G Security; "Specification of the MILENAGE Algorithm Set, General (Release 6)". *www.3gpp.org*.

[13] 3GPP, TS 35.201 V5.0.0 (2002-06), 3G Security; "Specification of the 3GPP Confidentiality and Integrity Algorithms (Release 5)". *www.3gpp.org*.

Management of Aggregation Networks for Broadband Internet Access in Fast Moving Trains

Frederic Van Quickenborne, Filip De Greve,
Filip De Turck, Ingrid Moerman, and Piet Demeester

Department of Information Technology (INTEC),
Ghent University - IBBT, Gaston Crommenlaan 8,
bus 201, B-9050 Gent, Belgium
Tel.: +32 9 33 14974, Fax: +32 9 33 14899
{frederic.vanquickenborne, filip.degreve}@intec.ugent.be

Abstract. Current satellite, GPRS and GSM systems lack from different shortcomings to provide broadband Internet access to fast moving trains. In this paper, we motivate that an Ethernet based aggregation network architecture is the best approach for realizing broadband Internet access in trains. The focus is on the management system for the Ethernet aggregation network and more specifically on the implementation of the management module for tunnel switching trigger management. The components of the management system are presented and three mechanisms for the implementation of the tunnel switching trigger management are detailed: a prediction based, a signaling based and a location aware approach. The different mechanisms are compared with respect to availability, complexity and network usage.

Keywords: Ethernet, Aggregation network, Trains, Prediction, Signaling, Location awareness.

1 Introduction

1.1 Motivation

Providing broadband Internet access to railway passengers is an interesting challenge. With the current emerging trials and early commercial releases it is only a matter of time before best-effort Internet on the train will become a reality. The satellite-based communication systems were the first solutions on the market but they lack uplink connectivity from train to satellite. These systems can combine satellite communication with GSM or GPRS receivers to ensure continuous up- and down-link coverage along the entire train trajectory [1]. Recently the first commercial bi-directional satellite communication system (4 Mbit/s down - 2 Mbit/s up) has been realized that can offer high speed Internet to high speed trains [2]. However as our own experiments on the Thalys trial Paris-Brussels show (see Table 1) the high satellite latencies make real-time communication impossible. This is due to the fact that the signal has to travel four times the distance Earth-satellite (twice for query and twice for answer). Currently an

Table 1. Main results from the Thalys Paris-Brussels experiment

Experiment	Result
Average experienced down-link bandwidth	ca. 1 Mbit/s
Average ICMP round-trip time	616 milliseconds
Average Voice over IP latency	4 seconds

alternative system architecture based on the WiMax pre-IEEE 802.16e standard is gaining interest: on-roof antennas with WiMax base-stations located near the railroad track provide a bi-directional broadband connection of 32 Mbit/s with seamless handoff in a high-speed environment [3]. In April 2005, the first broadband WiMax service on trains in the UK got operational [4]. Recently fast handoff between WiFi networks of roughly 5 milliseconds is established which promises near-seamless 802.11 roaming [5]. However, besides fast WiFi handoff the aggregation network which is responsible for the transport of data traffic from the fast moving users to the service providers' networks, has to be configured in-time and extended research is required on the management of the required tunnels in the aggregation network. The challenge is to design telecom networks in such a way that high bandwidth services which require a high level of Quality of Service – such as multimedia content delivery, video phoning and on-line gaming – can be provided. The focus in this paper is on the management of tunnel set-up and tear down triggers in an Ethernet aggregation network.

1.2 Ethernet Aggregation Network

The signaling protocols, used in this paper, are specifically implemented for Ethernet aggregation networks. This choice for Ethernet is motivated by the fact that telecom operators tend mainly for economical reasons towards networks consisting of standard QoS-aware Ethernet switches. Ethernet networks use a spanning tree protocol to maintain a loop free active topology. The legacy IEEE 802.1D [6] Spanning Tree Protocol (STP) and the IEEE 802.1w [7] Rapid Spanning Tree Protocol (RSTP), both use only N-1 links in a network of N nodes. This limits the amount of links that can be used in these networks but with the introduction of the IEEE 802.1s [8] Multiple Spanning Tree Protocol (MSTP) the bandwidth efficiency can be improved by maintaining multiple trees instead of a single tree. Most commercial switches are IEEE 802.1q & p compliant: i.e., they support the Virtual LAN (VLAN) technology and are QoS-aware (based on priority scheduling). VLANs provide a way of separating the physical topology in different logical networks and can be used to define end-to-end tunnels in the network. The configuration of VLANs can be performed automatically by means of the standardized GVRP (GARP VLAN Registration Protocol).

In summary, the ease of use and the auto-configuration of standard Ethernet, in combination with the recent advances in QoS support are probably Ethernet's strongest features.

The remainder of the paper is structured as follows: Section 2 presents the considered network architecture, whereas Section 3 details the implemented system for tunnel switching trigger management and elaborates on three trigger mechanisms. The management system design considerations are addressed in Section 4. Finally, the main conclusions are summed up in Section 5.

2 Considered Architecture

2.1 Aggregation Network

The FAMOUS (= FAst MOving USers) network architecture has already been published by the authors [9] and is depicted in figure 1. As can be seen in this figure, WiMax base-stations are positioned along the railroad track and every base-station is connected to an AGW (Access GateWay) which gives access to the aggregation network part. Note that in the aggregation network part traffic demands from separate users are not considered but groups of users are aggregated together per carriage. By aggregating the requests the system remains scalable for aggregation networks of realistic size. The traffic of each group of moving users is multiplexed in the AGWs into a VLAN tunnel. The aggregation network is responsible for the transport of the data traffic, by means of high bandwidth tunnels moving at high speed, to the service provider (SP) domain. The connection between the SPs and the aggregation network is realized by Service Gateways (SGWs).

2.2 Train Network - Aggregation Network Interaction

While commuters are moving along the railroad trajectory, their attachment point to the aggregation network will hop from one AGW to another. In order to preserve the connection between the train and the core network, tunnels must move with the trains. Due to the moving tunnel concept, seamless connectivity is not assured. However, service guarantees can be assured by making on-time resource reservations in the aggregation network. This prevents high congestion levels which are inherently harmful for the network performance during tunnel switching. There can be multiple connections per train, dependent on the number of antennas on the roof of the train. In this paper we assume that every carriage has its own antenna and that it is using a single associated tunnel in the aggregation network. These tunnels on which the data connections of fast moving users will be mapped, are VLAN-based tunnels which are responsible for the delivery to the correct AGW in the aggregation network. The VLANs are fixed end-to-end tunnels, automatically installed with GVRP. At their due time, tunnel reservations are registered for every connection, but only shortly before the connection will be effectively using the VLAN tunnel. When the train is no longer connected to the AGW and tunnel

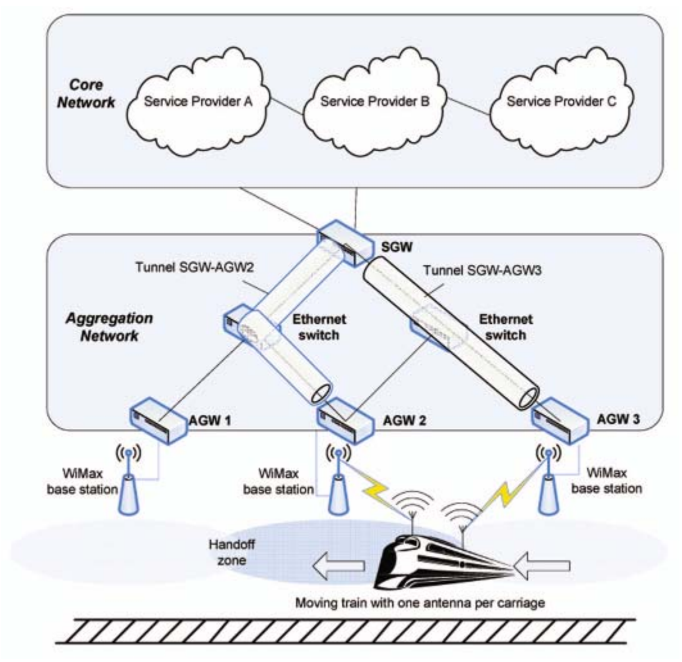


Fig. 1. Schematic representation of considered network architecture, which consists of a core part, an aggregation part and networks on the trains

reservations are no longer required, reservations are immediately released. In this way the system will always guarantee that the current and the next hop tunnel will be able to maintain the service level and that useless reservations are prevented. In a reservation-based system congestion levels and statistical multiplexing gains are manageable at the Connection Admission Control (CAC) level. In order to automatically activate the tunnels at their due time, a Layer 2 reservation protocol is developed, i.e. the GARP protocol G2RP (more details given in [9]). The designed protocols allow to make optimal use of VLANs (Virtual LANs) to support Multiple Spanning Trees in the Switched Ethernet networks. In this way network resources can be optimally used.

3 Management System

3.1 Overview

This paper focuses on the specific management framework for tunnel switching trigger, taking into account the exact positions of the trains. A diagram of a management system for aggregation networks to support fast mobile users has already been presented in [10]. Figure 2 shows the resulting management system. The off-line process has remained almost the same, but the online part is extended with new components. Two of them are essential: (i.) the triggers from trains and (ii.) the monitoring component. The first one collects information of trains (more specif-

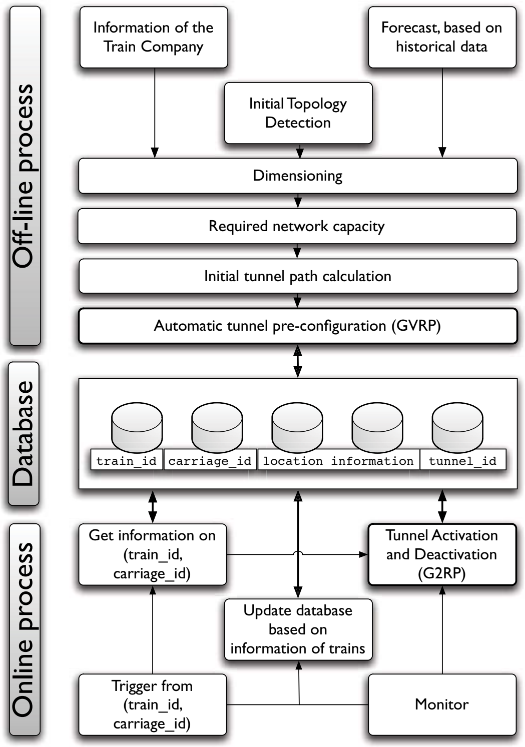

Off-line process
Database
Online process
Information of the Train Company
Forecast, based on historical data
Initial Topology Detection
Dimensioning
Required network capacity
Initial tunnel path calculation
Automatic tunnel pre-configuration (GVRP)
train_id
carriage_id
location information
tunnel_id
Get information on (train_id, carriage_id)
Tunnel Activation and Deactivation (G2RP)
Update database based on information of trains
Trigger from (train_id, carriage_id)
Monitor

Fig. 2. Schematic overview of the management system. It consists of two parts: an off-line part and an online part. The off-line process is done before any trains are moving on the rail tracks. Its main purpose is the pre-configuration of the needed tunnels. The different components of the off-line part are published by the authors in [11]. The second process is the online one. The main purpose of this part is to activate the pre-configured tunnels at their due time and deactivate them when they are not needed anymore.

ically information of each carriage of trains) and sends them to the component that gets the relevant information from the tables based on the received triggers. Using this information, the decision is made if a new tunnel must be activated or if an existing one must be deactivated. The monitoring component collects information on the usage of the tunnels. Based on this information, the management system is completely sure the new tunnel is used before deactivating the previous one. The central component uses all the provided information to update the tables, hence the tables contain most recent information.

3.2 Tunnel Switching Trigger Mechanisms

The management system supports three different tunnel switching mechanisms: (i.) a prediction based, (ii.) one based on signaling and (iii.) an exact location aware approach. The following sections describe each of these mechanisms.

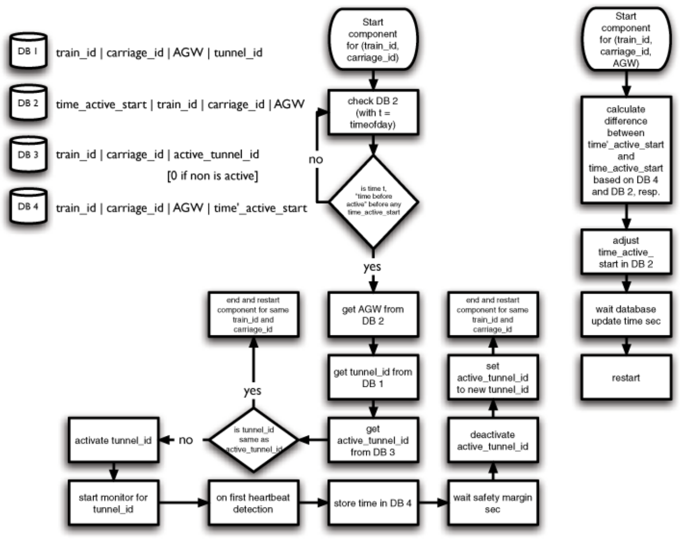

DB 1 train_id | carriage_id | AGW | tunnel_id
DB 2 time_active_start | train_id | carriage_id | AGW
DB 3 train_id | carriage_id | active_tunnel_id [0 if non is active]
DB 4 train_id | carriage_id | AGW | time'_active_start
Start component for (train_id, carriage_id)
check DB 2 (with t = timeofday)
is time t, "time before active" before any time_active_start
no
yes
get AGW from DB 2
get tunnel_id from DB 1
get active_tunnel_id from DB 3
is tunnel_id same as active_tunnel_id
end and restart component for same train_id and carriage_id
activate tunnel_id
start monitor for tunnel_id
on first heartbeat detection
store time in DB 4
wait safety margin sec
deactivate active_tunnel_id
set active_tunnel_id to new tunnel_id
Start component for (train_id, carriage_id, AGW)
calculate difference between time'_active_start and time_active_start based on DB 4 and DB 2, resp.
adjust time_active_ start in DB 2
wait database update time sec
restart

Fig. 3. Flowchart of the train position prediction mechanism. It uses tables based on information of the train company: the timetables and the velocity maps. Another process keeps the tables up-to-date based on the gained information.

Train Position Prediction - An overview of the mechanism is shown in figure 3. It is based on the exact timetables of the train company, together with a velocity map of the train speed on each part of the rail track system. Based on this information, a table is filled with time intervals and AGWs, together with `train_id`s and `carriage_id`s. The `train_id`s and `carriage_id`s are used to make a distinction between the different trains and their different carriages. A query on this table, at a certain moment in time, with a `train_id` and a `carriage_id` will result in a AGW, namely the AGW to which the carriage of that train is connected to, at that specific moment in time. Because trains can suffer from delays, the train position prediction mechanism has been extended with a heartbeat mechanism. Because the heartbeat signals, sent by a specific heartbeat component installed on every carriage, are sent directly to the management system, it allows for easy monitoring. Those signals are sent with a specific time between them, this is called the "heartbeat time" period. When a train is expected to reach a new base-station connected to an AGW, the system will check if the required tunnel has not been activated already. If this is not the case, the new tunnel is activated. Next the monitor detects the first heartbeat of the activated tunnel and uses this time to update the `time_active_start` of the table containing the expected arrival time of the carriage in the area of next AGWs. This is controlled by an external process, also shown in figure 3. After the monitor detected the first heartbeat through the recently activated tunnel, the previous active tunnel will be deactivated, after a pre-defined "safety margin" time period. This prevents the deactivation of a tunnel that is still in use.

Signaling - The second mechanism to keep track of the exact positions of the carriages, needs an action taken by the base-station. The moment the base-station detects an antenna of a carriage arriving in its covered area, it sends the `train_id` and `carriage_id` of the antenna to the AGW connected to the base-station. This AGW broadcasts a message including the `train_id` and the `carriage_id`. From the source Ethernet address the SGW knows the origin AGW of the message. From a lookup in its table, the correct `tunnel_id` is retrieved. Next the management system checks if the the found `tunnel_id` is activated already from another table (the one that holds the active `tunnel_id` for each `train_id-carriage_id` pair). If this is not the case, the `tunnel_id` is activated. After a specific time, the previously activated tunnel to that `carriage_id` is deactivated, after a pre-defined "safety margin" time. If no tunnel has been activated before for the considered `carriage_id`, deactivation is not needed. Finally the table containing the active `tunnel_id` for the considered carriage is updated with the new value. Because the speed of the train can be very high, and because the signaling process takes some time to activate the tunnel, it can be useful to handle a dedicated antenna to signal the presence of the train. This antenna is placed on the first carriage of each train. This antenna will not be used for the data traffic, but only for the signaling process.

Location Aware Approach - The third mechanism uses, just like the previous one, information obtained from the train. However, this information will be updated at regular moments in time. Therefor a specific "update time" must be set. The location aware components on the carriages send their information every "update time" seconds. For the location information, GPS modules are used, as they are more and more often installed on trains. With this information it is possible to know which carriage is in reach of which AGW, and so it is known what `tunnel_id` must be activated, from the appropriate table. As it is the case in the previously described signaling mechanism, the deactivation of the previous tunnel is done after a "safety margin", if there is any previous active tunnel.

4 Design Considerations

For each of the proposed mechanisms different constants must be defined, several variables must be declared and tables must be implemented. Table 2 gives an overview.

4.1 Message Sequence Charts (MSC)

- Train position prediction - In figure 4 the MSC is given for the train position prediction mechanism when a train is on schedule. It is shown that after activation of the new tunnel, the monitoring is started. After detection of the first heartbeat through the new tunnel, a safety margin is waited and the old tunnel is deactivated. As depicted, the new tunnel is ready when the first data is sent through the tunnel. Also the release of the old tunnel is done in a proper way.

Table 2. Summary of the design considerations

Type	Name	Prediction	Signaling	Location
Constant	safety margin	x	x	x
	time before active	x		
	heartbeat time	x		
	table update	x		
	location update			x
Variable	first activity	x		
Table	with `tunnel_ids`	x	x	x
	with `active_tunnel_id`	x	x	x
	with estimated `time_active_start`	x		
	with measured `time_active_start`	x		
	with GPS information			x

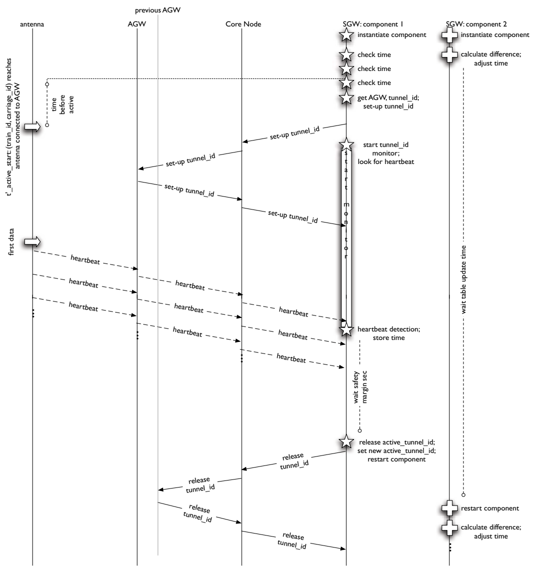
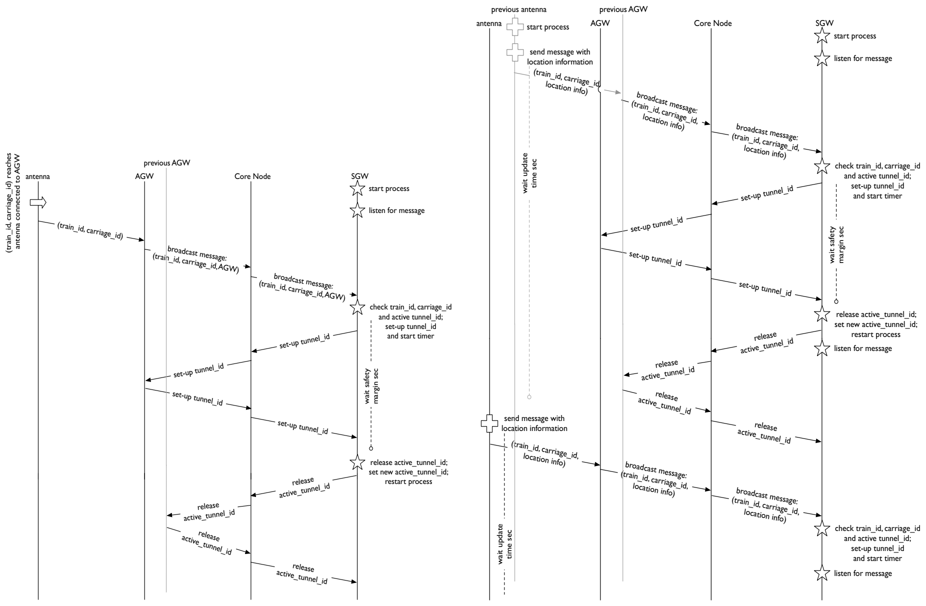


Fig. 4. The MSC shows both components that are instantiated at the SGW. Several parameters are indicated: the time before active, the safety margin and the table update time.

– Signaling - Figure 5(a) shows the MSC for the signaling mechanism. It shows the signal on which the tunnel activation is based. After receiving this signal, the system waits a predefined time, the ”safety margin”, and afterwards deactivates the old tunnel.
– Location aware approach - Finally, figure 5(b) shows the MSC for the location based train position detection. It is clear that, based on the knowledge of almost the exact position of the trains, all needed tunnels will be activated when they are needed. As disadvantages of this system, we mention


(train_id, carriage_id) reaches antenna connected to AGW
antenna
previous AGW
AGW
Core Node
SGW
start process
listen for message
(train_id, carriage_id)
broadcast message: (train_id, carriage_id,AGW)
check train_id, carriage_id and active tunnel_id; set-up tunnel_id and start timer
set-up tunnel_id
wait safety margin sec
release active_tunnel_id; set new active_tunnel_id; restart process
release active_tunnel_id
previous antenna
send message with location information
(train_id, carriage_id, location info)
broadcast message: (train_id, carriage_id, location info)
wait update time sec

(a) MSC for signaling mechanism. (b) MSC for location based mechanism.

Fig. 5. MSC for different mechanisms

the purchase and installation of GPS infrastructure on trains and network usage by the broadcast messages with the GPS location information inside.

4.2 Comparison

For the comparison of the different approaches, we compare the three proposals on their availability, their complexity and their efficiency. These comparisons are given in tables 3 and 4. The first table compares the quality of the connection during tunnel switching in three cases: if the train is on schedule and during the detection of a small or a large delay. In the first table the influence of the used technology can not be neglected, as this determines the distance between the successive base-stations and this has consequences on the availability of the connection. If the system acts slowly, and the time between two handoffs is small, the system will fail. If the distance between base-stations is small, this failure will be dramatic, for bigger distances, the problem is not necessarily dramatic. This is why the WiFi technology is worse than the WiMAX technology with respect to the quality of connection. This quality also depends on the amount of delay, but only for the prediction based method: for small delays, a small number unnecessary reservations will be made, for larger delays, this amount becomes bigger, leading to an even worse quality of connection. As a summary, following observations can be made: (i) if a management solution is desired,

Table 3. Comparison of different approaches concerning availability. When using the WiFi technology, the base-stations are placed very close to each other. The WiMAX technology permits larger distances between base-stations while keeping the overall bandwidth equal. We like to remark that for both cases (WiFi and WiMAX) all parameters are the same e.g., the location update time is equal.

	Prediction		Signaling		Location	
Train	WiFi	WiMAX	WiFi	WiMAX	WiFi	WiMAX
On time	++	++	−−	−	+	++
Little delay	−	+	−−	−	+	++
Large delay	−−	−	−−	−	+	++

Table 4. Comparison of three approaches concerning complexity and network usage. The complexity is measured in terms of the number of required tables, variables and constants, together with the required number of component instantiations. The network usage is determined by the number of messages for tunnel set-up and tear down.

Metric	Prediction	Signaling	Location
Complexity	−	++	+
Network usage	++	+	−

which is transparent to the existing base-stations and managements components on trains, the prediction approach is the only possible solution; (ii.) although, if the complexity of this prediction approach is unaffordable, a signaling or location aware solution is preferred; (iii.) a location aware approach has the advantage that tunnels are always established in time at the expense of the more complex GPS management components.

5 Conclusion

As a reaction on the existing technologies, which suffer from different kinds of problems to provide broadband Internet access to railway passengers, we proposed a new approach, based on an Ethernet aggregation network architecture. This paper focused on the management system for the tunnel switching trigger generation. Several mechanisms for the management and generation of the necessary triggers have been elaborated on. The weaknesses and strong points of the different mechanisms have been addressed. The involved software components and database structure have been detailed, together with a motivation for the implementation choices.

Acknowledgment

Research funded by PhD grant for Frederic Van Quickenborne (IWT-Vlaanderen) and by postdoc grant for Filip De Turck (FWO-V).

References

1. Icomera wireless onboard internet, web site. *http://www.icomera.com.*
2. Broadband internet access on train, web site. *http://www.21net.com.*
3. Wi-lan launches libra mobilis, press communication. *http://www.wi-lan.com/news/press/20041019.htm.*
4. P. Judge. 100 mph wimax hits the rails to brighton. *http://www.techworld.com/mobility/features/index.cfm?FeatureID=1351*, April 2005.
5. Ramani I. and Savage S. Syncscan: Practical fast handoff for 802.11 infrastructure networks. *Proceedings of the IEEE Infocom Conference 2005*, March 2005.
6. IEEE 802.1D. Standards for local and metropolitan area networks: Media access control (mac) bridges. 1990.
7. IEEE 802.1q. Standards for local and metropolitan area networks: Virtual bridged local area networks. 1998.
8. IEEE 802.1s. Standards for local and metropolitan area networks: Multiple spanning trees. 2002.
9. F. De Greve, Van Quickenborne F., and et al. Famous: A network architecture for delivering multimedia services to fast moving users. *To appear in Special Issue of the Wireless Personal Communications Journal, Springer Publishers*, 2005.
10. Frederic Van Quickenborne, Filip De Greve, Filip De Turck, and Piet Demeester. On the management of aggregation networks with rapidly moving traffic demands. *IFIP/IEEE International Symposium on Integrated Network Management*, 2005.
11. Frederic Van Quickenborne, Filip De Greve, Filip De Turck, Ingrid Moerman, Bart Dhoedt, and Piet Demeester. Optimization models for designing aggregation networks to support fast moving users. In Gabriele Kotsis and Otto Spaniol, editors, *EuroNGI Workshop*, volume 3427 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 2004.

Design and Implementation of an Open IMS Core

Dragos Vingarzan[1], Peter Weik[1,2], and Thomas Magedanz[1,2]

[1] Technical University of Berlin,
Faculty IV / Next Generation Networks,
Franklinstr. 28-29, 10587 Berlin, Germany
http://www.av.tu-berlin.de

[2] FOKUS Fraunhofer-Institute for Open Communication Systems,
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
{vingarzan, weik, magedanz}@fokus.fraunhofer.de
http://www.fokus.fraunhofer.de/

Abstract. Mobile network operators in Europe mostly have rolled out 3G networks and are now looking into the deployment of the IP Multimedia Subsystem (IMS) which provides an overlay architecture for IP based core networks. The IMS offers a standardized way to develop new and appealing applications and may serve as a blueprint for convergent Next Generation Networks. As with the introduction of any new technology, there comes the need to measure its performance and capabilities.

The Open IMS Core project introduced in this paper offers the core network functionality of an IMS for testing the performance of the central IMS network entities, the IMS Call Session Control Functions. The project's target was to build an implementation based upon open source tools that is fully conformant to existing IMS standards which may be re-used in other IMS related projects. In its setup the Open IMS Core resembles future IMS networks at the needed detail level and allowed to produce first IMS performance results.

Keywords: IMS,NGN,CSCF,SIP,Performance.

1 Introduction

With the installation of 3G technologies mobile network operators broadened the data path for their customers but so far haven't exploited it. IMS aims to fill this void since it promises to enable bundled mobile multimedia services enriched with presence and location information in a quality of service assured manner. IMS is standardized by the 3rd Generation Partnership Project (3GPP) based on specifications and protocols of the Internet world. It is structured in a way that it will provide almost unlimited multimedia capabilities for customers plus a very flexible service layer while offering appealing service creation and delivery possibilities for the operators.

The main IMS signaling is based upon the Session Initiation Protocol [1], a general dialog initiation protocol between two or more peers. From the technical point of view, SIP is a HTTP-like protocol based on requests and responses between nodes, capable of carrying the signaling of arbitrary contents. It emerged in the Voice over IP (VoIP) world (but it is not limited to it) as a response to the complicated stack of H.323 solutions. Besides SIP, the IMS uses other IP-based protocols like Diameter [2] or H.248 [3] to provide a core network architecture which supports the integration of existing IN services with new SIP-based ones.

Since mobile devices are becoming smarter and are constantly gaining in processing power, the SIP and IMS specifications moved a part of the signaling load from the core, as in a SS7 environment, to the edge of the network. The shift of a mobile network setup from a model where a rather dumb client communicates with an intelligent network towards the model of the Internet where a smart client communicates with a network that merely ensures routing[1] is what makes the IMS a catalyzator for convergent services.

But as with any new technology, first it has to be proven that it works and scales well before we will see a big deployment. SIP has proven that it is able to handle high traffic load in existing VoIP networks with limited subscriber numbers but will it be able to perform just as well as the current core network technology of a mobile network operator ?

This paper will describe a protoypical implementation of core elements of an IMS network, the *Open IMS Core* and its application in a first performance test. The next chapter describes the architectural design of the Open IMS Core, while chapter 3 will highlight some implementation details of the architecture. Chapter 4 outlines the performance of the Open IMS Core which was successfully tested in the Open IMS playground [4] at the Fraunhofer Institute FOKUS. A brief summary finishes the paper.

2 The Open IMS Core Components

Figure 1 shows a simplified architecture of an IMS network as specified in [5]. We regard the IMS core network elements to be the ones that are utilized to route the SIP signaling for establishing, updating or ending multimedia sessions.

Due to the decentralized nature of the network, routing has a big percentage of the total processing power. Endpoint peers exchange and process many SIP messages during the setup and termination of their dialogs. The proxies must handle all the load with low latency to ensure a low total dialog setup time. This task is fullfilled in the Call Session Control Functions (CSCFs) of an IMS. They are therefore the main part of the Open IMS Core. For them there is the need for registrar services that keep track of the users, their preferences and their data. They are part of the CSCFs for user location, user enabled services and firewalling the IMS core from possible attacks. These types of registrar services are part of the core network as they allow the retrieval of the correct destination for message routing. Finally the IMS core must also be capable to act as endpoint in the signaling path for certain scenarios like forced call termination or back

[1] and in case of IMS adds quality of service on top of it

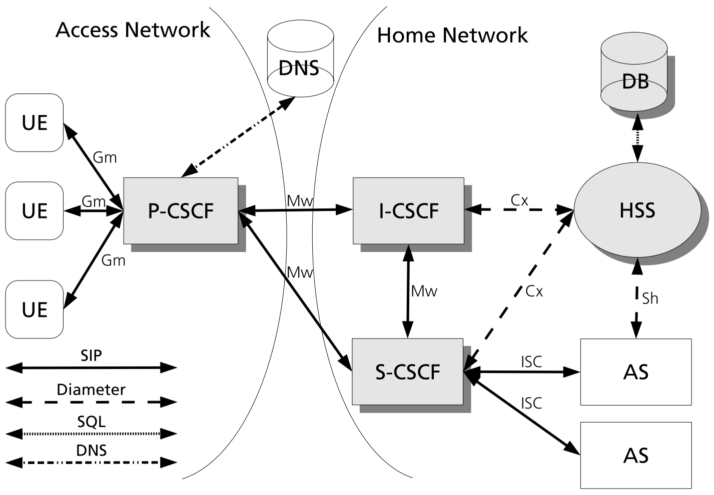


Fig. 1. The Open IMS Core in a simplified architecture of an IMS network

to back (B2B) calls. Additionally, since the proper functionality of the CSCFs relies on service specific user profile information and a lookup functionality of user assigned CSCFs, a Home Subscriber Server (HSS) is also part of the Open IMS Core.

Presence or Grouplist Management services as well as any other generic ones are implemented into the various Application Servers (AS) that can participate actively in the signaling and media paths but they are not considered to be part of the IMS core network. Neither do the User Endpoints (UE) since they generate signaling traffic.

In a first step, the Open IMS Core was required to enable the IMS compliant setup of multimedia calls between UEs to produce benchmarking results. Therefore it was optimized for routing IMS SIP traffic between two mobile phones rather than supporting specific IMS services hosted on application servers.

2.1 Open IMS Core Network Elements

CSCFs - The Call Session Control Functions are entities which are routing and processing the actual SIP signaling. Their functionality is specified as rules of SIP processing. Categorized by functionality, there are three CSCFs entities: Proxy, Interrogating and Serving CSCF.

1. Proxy - Call Session Control Function (P-CSCF). The P-CSCF represents the entry point in the IMS network. The access network responsible for data transfer at the IP level will transparently provide for the UEs access to the P-CSCF. The functionality contains all three possible patterns of a SIP entity (proxy, registrar and B2B user agent).

2. Interrogating - Call Session Control Function (I-CSCF). The I-CSCF is a SIP proxy capable of finding the location of a serving CSCF (S-CSCF) of a user, if he/she was registered or determines where the user should be registered during the registration process. By sending a Diameter request to the HSS and processing the answer it retrieves the user's current S-CSCF and then forwards the SIP message there. It is the entry point in the home network, and must be a fast entrance redirection dictionary. The functionality of the I-CSCF contains only the SIP proxy pattern.
3. Serving - Call Session Control Function (S-CSCF). The S-CSCF is the work-horse among the CSCFs. It acts in all three functional patterns: registrar for users, B2B user agent for call control and proxy for forwarding the messages to their destinations or through the application servers. It has to apply the specific routing mechanisms for SIP messages that are downloaded during registration in the user's profile.

HSS - The Home Subscriber Server is the data storage entity within an IMS. It can be seen as the evolution of the Home Location Register in all-IP core networks. It stores the user profile, provisioning data and keeps registration and location status information of all subscribers. The HSS provides the information to application servers and CSCFs over Diameter-based reference points (e.g. the one defined in [6]).

2.2 Additional IMS Network Elements

UE We needed some User Endpoints that were ready to perform an IMS registration process and to handle the SIP signalling of next generation mobile phones. The UEs had to be capable of supporting the IMS Authentication and Key Agreement (AKA) [7, 8] and had to fullfill SIP IMS specific requirements like special message routes or subscription to its own registration event.

AS - Application Servers allow to demonstrate specific service usage on top of the Open IMS Core. Yet, their demonstration is not within the scope of this paper. Any third party AS conforming to the IMS SIP standards [9, 10] can interoperate correctly with the core through the SIP-based IMS Service Control (ISC) interface.

3 Implementation

The actual realization of an IMS core network proved to be a difficult task. The main challenge was that it had to perform very well in speed and in functionality. It had to reduce the processing to the minimum while it implements all functionalities that are required in the specifications even if they were not tested in the specific test setup which leaves Application Servers out of the signaling path. We were trying to have a reference implementation by closely following the specifications and not compromising conformity for performance. For all interfaces related to the Open IMS Core we followed the specifications for Release 6 of the 3GPP IMS.

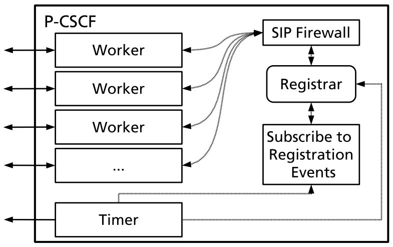


Fig. 2. Implementation of the P-CSCF

3.1 CSCFs

The CSCFs are built upon the SIP Express Router (SER) [11] which can act as SIP registrar, proxy or redirect server and is capable of handling many thousands of calls per second. It is an open source SIP server implemented in the programming language C and has a modular structure that permits to make functionality additions. It successfully runs on many Unix-based platforms like Linux, BSD or Solaris. Each CSCF entity of the Open IMS Core is implemented as a SER dynamically loadable module that adds the required operations to the SER so that it can act according to the specific 3GPP technical specifications. The modules are capable of multi-threaded processing and can keep supplementary state information. There is a special focus towards scalability for both, load distribution and data quantity.

Proxy-CSCF. In the current implementation of the Open IMS Core, the P-CSCF component is able to firewall the core network at the application level: only registered endpoints are allowed to insert messages inside the IMS network and the P-CSCF asserts the identity of the users. To keep track of the registered users, it has an internal registrar that is updated by intercepting the registration process and later by subscribing to the registration package at the S-CSCF and receiving notifications. The actual data is kept as a hash-table to allow fast retrieval. For originating call signaling it generates unique charging vectors and inserts network and path identifiers that are needed for the correct further processing of the SIP messages. UE forged information that might lead to an attack is removed. After a successful registration process to an IMS home network, subsequent user messages are forwarded based on DNS information towards the requested IMS network. In the outgoing direction, towards the user endpoints, it can act as a NAT router by simply being active in both networks. After user specific security data, like cypher keys has been eliminated, the messages are forwarded directly to their destination. Filtering in this direction is needed since cipher material must not be sent over the potentially untrusted connection between the UE and P-CSCF.

Interrogating-CSCF. The I-CSCF has the role of a stateless proxy that, by using the indicated public identities of the caller or the callee, queries the HSS and

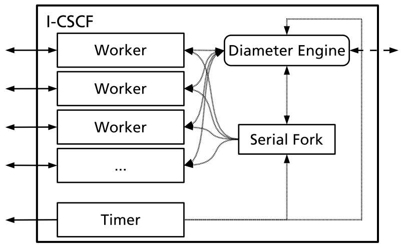


Fig. 3. Implementation of the I-CSCF

based on its response routes the message to the correct S-CSCF. It implements the Cx interface of an I-CSCF to the HSS. Therefore it supports the required Diameter commands to locate the user-assigned S-CSCF and to assign a new S-CSCF and check identities and roaming authorizations as specified in [6].

After receiving a successful answer for the Diameter query the I-CSCF forwards the SIP messages in a transactional mode. It is optimized for speed and no state information is kept. To protect the home network, it has a firewalling capacity that only allows messages coming from trusted networks.

Serving-CSCF. The S-CSCF implementation also communicates with the HSS using Diameter (over the Cx interface) to retrieve authentication vectors, update registration information and download the user profiles as specified in [6]. Eventhough it was not relevant in our current tests, the S-CSCF can apply the user profile based filter criteria to enforce specific SIP routing rules. It implements support for carrying out the IMS Digest AKA version 1. Rather than generating authentication vectors it relies on the HSS for this task and compares these values to the ones calculated in the UE. The registrar of the S-CSCF has a complex structure as a hash-table. The information that is needed to relate an user identity to a physical user endpoint is stored here and used furtheron for call routing. It also accepts subscriptions to registration states and notifies the subscribers of changes in the registrar.

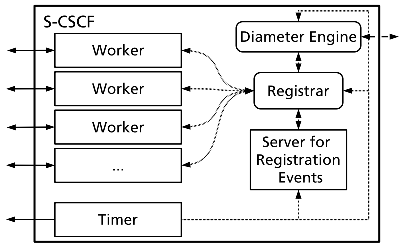


Fig. 4. Implementation of the S-CSCF

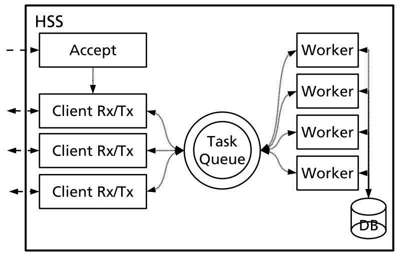


Fig. 5. Implementation of the HSS

3.2 HSS

As its purpose in the Open IMS Core is that of a database, the implemented HSS is targeted towards fast performance. It is merely a glue between a database management system (DBMS) and the Diameter interface with the CSCFs. Protocol checks and Diameter command logic is implemented here, but it could just as well be implemented as stored functions in the DBMS. The currently supported Diameter commands of the HSS therefore are only the ones used in setting calls up rather than enforcing changes in authorization or user profile. Additionally, the HSS allows the generation of authentication vectors. As the primary requirement to the HSS is speed, it is implemented in C. It has a multi-threaded architecture, capable of handling many clients and processing the requests through parallel workers. It does not keep any IMS state in memory but uses the database for this purpose. The user data is kept inside a MySQL database. So far the main bottleneck is the database access. Caching or other optimization methods are not efficient as the DBMS is already implementing that.

4 Performance Evaluation of the Open IMS Core

In order to verify the functionality of the Open IMS Core, we used three different "traffic generators". In addition to developing an in-house solution of a UE and adopting an existing open source SIP client, the Fraunhofer Institute FOKUS team especially extended a SIP conformance testing platform from pure IETF SIP towards IMS traffic generation. We therefore successfully tested with the following SIP clients:

1. Open SIP Client - The Open SIP Client of the Fraunhofer Institute FOKUS is written entirely in Java and runs on supporting current mobile phones enabling them to act as a full 3Gbeyond phone.
2. KPhone - The open source SIP client KPhone [12] was modified for use in an IMS environment. It is capable of instant messaging, audio and video calls as well as for signaling user presence.

3. Benchmarking Simulated UE - For benchmarking the Open IMS Core, we needed a test system capable of loading it with traffic. The Fraunhofer team specified abstract testing suites in the Testing and Test Control Notation (TTCN-3) which are applied in the Test Lab for simulating the behavior of many IMS subscribers in setting up calls and measuring associated QoS parameters. We used a specially adopted *TTworkbench Enterprise Edition* of Testing Technologies [13] to carry out our performance tests.

Performance testing on a telecommunication platform implies the simulation of scenarios in the most realistic model possible. To measure the performance of the Open IMS Core in setting up multimedia (voice or video) calls, we first defined in the following heuristic approach a target load of the system deducted from the total number of subscribers that the core network should serve.

– For a *Total Number of Subscribers (TNS)*, at any moment there is certain *Percentage of Registered Subscribers (PRS)* in the network. This means that we assume to have at any given time a *Number of Registered Subscribers (NRS)*: NRS = TNS x PRS.
– For the number of registered subscribers that are online there will only be a certain percentage of them engaged in a call. We are defining this as *Percentage of Active Calls (PAC)*. The *Number of Parallel Calls (NPC)* to handle is then NPC = NRS x PAC.
– A measurable number in our system is the number of *call-setups attempted per second (CPS)*. To determine this from the number of parallel calls we would need to know the *Average Call Time (ACT)*. The CPS then indicates how many calls have to be initiated or terminated per second: CPS = NPC/ACT.
– To summarize:

$$CPS = \frac{TNS \times PRS \times PAC}{ACT} \tag{1}$$

To set up a target performance goal, we were considering to handle a fictive network of 100.000 subscribers. We have choose real-life values to the best of our knowledge with 1/3 of the subscribers "online" and registered simultaneously and 1/16 of them engaged in a multimedia call with a average duration of 180 seconds. This meant that our system would have to be able to sustain a load of 11.57 calls per second. Since one successful end-to-end multimedia call setup within IMS requires about seven SIP messages, the Open IMS core would have to be able to process a minimum of 81 SIP messages per second. In our test setup we used one workstation with a Pentium 4 processor running at 3GHz with HyperThreading activated. While running all the four components of the Open IMS Core and the DBMS it was able to sustain a load of about 120 SIP messages per second. At the maximum load the tested system was able to handle at least 17 calls per second, a number well beyond the targeted speed and still within the set latency boundaries of 3 seconds (see figure 6).Pushing the system above a 300 SIP messages limit with a script-based load generator resulted in high delays and further into a cascading effect that blocked proper functionality.

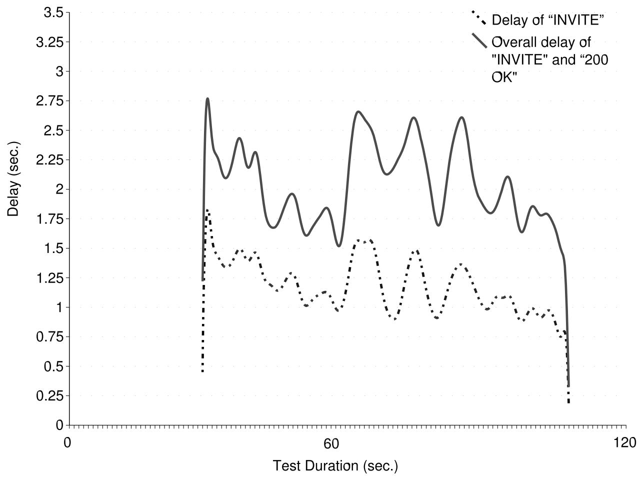


Fig. 6. Delay of Calls (incl. en- and decoding of SIP messages within the test platform)

We must mention that in this first simple performance test we have used just one instance of each CSCF and respectively a rather simple HSS. In real world scenarios operators probably will be using multiple instances for reasons of load-balancing and fail-safety. Additonally, the methodologies and the test system to measure the performance of IMS core elements will be refined in future work towards expected real-life conditions for IMS installations.

5 Summary

This paper described the design and implementation of an IMS core network that was done at the Fraunhofer Institute FOKUS in the beginning of 2005 which eventually will be released as Open Source project. We outlined the functionalities of the core IMS network elements and gave some triggers on the implementation of which the SIP session control functionality is based on a well recognized and proven SIP server solution, the SIP Express Router. In combination with a built-in HSS connection, the Open IMS Core is able to perform a 3GPP IMS compliant user registration and routing of SIP traffic for multimedia signaling. We described a simplified heuristic performance test model that enabled us to carry out a first assessment on what kind of performance can be expected for SIP signaling in coming IMS installations.

References

1. J. Rosenberg, H. Schulzrinne et. al., "SIP: Session Initiation Protocol", RFC 3261, June 2002
2. P. Calhoun, J. Loughney et. al., "Diameter Base Protocol", RFC 3588, September 2003

3. C. Groves, M. Pantaleo et. al., "Gateway Control Protocol Version 1", RFC 3525, June 2003
4. The FOKUS IMS playground - http://www.fokus.fraunhofer.de/ims
5. 3GPP TS 23.228 - 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS); Stage 2 (Release 6)
6. 3GPP TS 29.228 - 3rd Generation Partnership Project; Technical Specification Group Core Network; IP Multimedia (IM) Subsystem Cx and Dx interfaces; Signaling flows and message contents (Release 6)
7. 3GPP TS 33.203 - 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G security; Access security for IP-based services (Release 6)
8. A. Niemi, J. Arkko, V. Torvinen, "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)", RFC 3310, September 2002
9. 3GPP TS 24.228 - 3rd Generation Partnership Project; Technical Specification Group Core Network; Signaling flows for the IP multimedia call control based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 (Release 5)
10. 3GPP TS 24.229 - 3rd Generation Partnership Project; Technical Specification Group Core Network; IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage3 (Release6)
11. The SIP Express Router - http://www.iptel.org/ser
12. Wirlab Network Research Center; KPhone - http://www.wirlab.net/kphone/
13. Testing Technologies - http://www.testingtech.de/

Mobility-Aware Coordination in a WLAN Hot-Spot Area

Karin Anna Hummel

Institute of Distributed and Multimedia Systems,
University of Vienna, Lenaugasse 2/8, A-1080 Vienna
karin.hummel@univie.ac.at

Abstract. *Mobile computing* is a natural further development of distributed computing enabled by wireless networks, portable devices, and recently wearable computers. Due to varying wireless network conditions while moving distributed computing is facing new challenges. In such scenarios, *data space-based* middleware offers promising mechanisms by supporting asynchronous coordination. However, reactive mechanisms provide no means to invoke operations which have to be started in advance, for example, before a mobile device looses connection.

This paper makes use of the *Mobility-Aware Coordination Layer* approach *(MobACL)* which extends the concept of shared data spaces by means of proactive fault tolerance mechanisms. Mobility-awareness is achieved by monitoring current wireless link states and by predicting future wireless link states and retention periods based on *mobility models.* Experimental results are presented which demonstrate the potential for increasing coordination operation throughput by means of the MobACL and replication of processes for four well-known *coordination patterns.*

1 Introduction and Related Work

Mobile devices and wireless networks allow mobile users to connect to Web-services, peer-to-peer services, and pervasive computing environments. Collaborations between mobile users are more and more exploited in application domains such as *health-care* and *emergency services* (e.g. for fire fighters). Additionally, ongoing research enhancements in the field of wearable computing envision application scenarios for distributed computing while a person is moving (with and without direct human computer interaction). Despite of the variety of available mobile technologies, the common ground of mobile computing lies in the dynamics due to *movement.* From the user's or mobile application's perspective, the effects of movement are often perceived as degradation of service quality caused by connection loss and decrease of wireless link quality.

Consequently, *coordination* of distributed tasks executing on mobile devices requires additional supporting technologies. Such mechanisms have already been proposed for middleware systems [17]. In order to support operations on shared resources while being disconnected, processes on mobile devices will often operate on copies and cached data. Early works, like the resource-centric work proposed in [19], have already identified major means for mobility support, that are, emulation of resources, caching, and use of local resources. Accuracy while creating the copy, efficient synchronization, and data lock management have been identified as key challenges [13] and are addressed by recent middleware approaches (see, for example [6]).

Data Space (DS) middleware approaches[1] provide persistent communication and coordination resources, like proposed by CORSO [16, 15] or JavaSpaces.[2] Participants may reconnect and retrieve information stored while they have been disconnected. However, times of disconnections are commonly not supported further. Issues related to working on copies, which are, (i) creation of the copy and (ii) synchronization when reconnecting based on versioning are discussed in [18].

The reactive mechanisms described are not sufficient for activities which have to be invoked before the network conditions worsen. Very recently, promising proactive mechanisms are proposed which use pre-fetching of events based on rule based mobility predictors [3, 8]. ScALADe [2] is another related approach which uses prediction for mobile agent migration based on uncertainty modeling. Related to these approaches, the author [12] has proposed proactive copying of shared data and releasing of locks in an earlier work by introducing the *Mobility-Aware Coordination Layer (MobACL)*. Here, proactive fault tolerance mechanisms make use of prediction of future link states and retention periods based on *mobility models*.

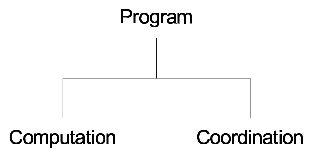
In this paper, the possibilities of increasing coordination operation throughput in a WLAN hot spot area are investigated for two mechanisms: (i) the MobACL approach *(proactive copying)* and (ii) *process replication*. Hereby, this paper contributes by experimentally investigating the consequences of mobility in different distributed application scenarios. In order to derive application independent results, these scenarios are modeled by means of *coordination patterns* and classified according to their coupling characteristics in terms of time and reference.

This paper is structured as follows: Section 2 introduces the coordination pattern approach and describes the four patterns under investigation. Section 3 presents the mobility-aware approach. In Section 4 the protypical implementation is described and in Section 5 the experimental results are presented.

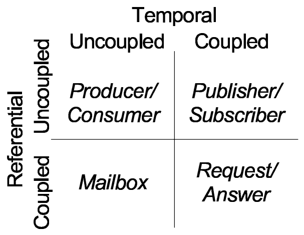
2 Coordination Patterns

In application domains like emergency services, persons in charge are often mobile and collaborating. In order to support coordination of, for example, fire fighters, health personnel, and military or police forces, the distributed workflows rely on *coordination* rules to exchange information and to control the activities. By separating such concurrent applications into parts related to either

[1] These approaches originate from the *Linda Tuple Space* [11].
[2] http://java.sun.com/



(a)



(b)

Fig. 1. (a) Concept of concurrent programming and (b) classification of distributed coordination

computation or coordination (Figure 1(a) [11, 4]), it is possible, to study coordination aspects separated from the computational problem. Similar to related works describing coordination scenarios [9, 20, 16, 5], this work uses *coordination patterns*, that are, software design patterns [10] for coordination purpose in order to model the interactions of distributed processes. Thus, it is possible to derive significant results for distributed mobile applications which exhibit the same coordination characteristics as modeled by the patterns. This work is further based on the *space based computing* (shared *data space (DS)*), where distributed processes communicate by writing and reading from the shared space instead of sending messages directly to one another.

Common taxonomies for concurrent processes which interact with one another refer to the aspect of *referential coupling* (or *spatial coupling*) and *temporal coupling* [21, 5]. The following definitions are used:

> A coordination pattern is *referentially coupled* if one or more participating processes perform at least one coordination operation which addresses another particular process explicitly. Otherwise it is *referentially uncoupled.*[3]

> A coordination pattern is *temporally coupled* if one or more participating processes perform at least one blocking coordination operation (that is, a participant waits for a message from another participant). Otherwise it is *temporally uncoupled.*

Based on a literature survey, for each class derived, a representative coordination pattern can be identified (Figure 1(b)):

Producer/Consumer. This pattern describes the interactions of two process types. The producer creates data items and adds them to the shared DS, while the consumer reads data items from the shared DS in a consuming manner. The processes execute their code asynchronously and without a

[3] Although in the shared DS messages are not sent to processes directly, identifiers can be used to address processes logically.

reference identifying the remote processes. Variants of the pattern can be derived by altering some pattern attributes (see also [14]).

Publisher/Subscriber. In this coordination pattern two process types interact with one another (see also [9]). A subscriber selects events of interest, waits until it is notified about the occurrence of such an event, and reads the new data. The publisher process type is responsible for producing data events. Notifications are generated by means of the shared DS. Note, that the processes are not addressed directly. The subscribers need to be informed in (soft) real-time (in fact, this process waits for notifications). This pattern can be used to synchronize states among distributed processes.

Mailbox. In the *Mailbox* (also known as *Mailbox Coordination* [21]) pattern, each process is of the same type - here, it is said to be a peer. A peer sends messages to other peers and fetches messages in an asynchronous manner (that is, writes to and reads from the shared DS). Each process can be addressed explicitly by a specific name or mailbox in the DS.

Request/Answer. The *Request/Answer* coordination pattern [14] refers to a typical client/server scenario, where the client initiates the communication by writing a service request to the shared DS. The client blocks until the server's answer is written to the DS using a name assigned to the client. *Client/Server* pattern is another name of this coordination pattern which is extended by a dispatcher used as a name service in [9].

3 Mobility-Awareness in WLAN Hot-Spot Areas

In a WLAN hot-spot area, processes executing on mobile computers communicate over wireless links. The *Signal to Noise Ratio (SNR)* sensed is a means for characterizing wireless link quality which determines the data transfer rates available [1]. While moving from one location to another, link quality changes. Figure 2 depicts a specific and simplified topology assuming one WLAN access point. By mapping link quality states to locations, movement can be interpreted as a transition from one WLAN link quality state to another.

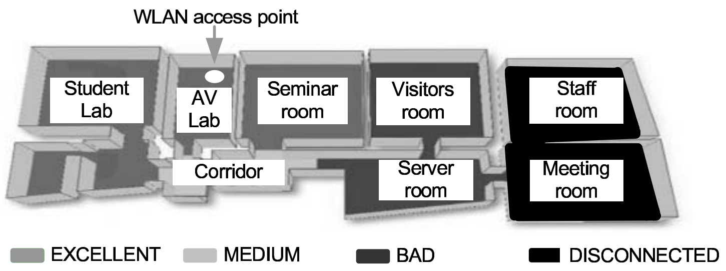


Fig. 2. Mapping of rooms to link quality states

3.1 Fault-Hypothesis

The approach presented assumes that wireless link degradation related to movement leads to failures of distributed applications. In link state *BAD* decreased coordination throughput is considered and in link state *DISCONNECTED*, no coordination activities are possible.[4]

The fault tolerance mechanisms applicable are: (i) release of data locks, (ii) working on copies, and (iii) process replication. While (i) applies for all distributed coordination pattern classes, (ii) only applies for temporally uncoupled coordination patterns and (iii) can be used in case of referentially uncoupled coordination patterns. After reconnection, synchronization operations take place.

3.2 Prediction

Prediction of next link states and retention periods based on mobility models [7] is proposed as a means to invoke proactive fault tolerance mechanisms. Two mobility models are currently defined, that are, the *Continued Move Mobility Model (Continued Move)* and the *Smart Office Mobility Model (Smart Office)*. Note, that the MobACL architecture is open to new mobility models.

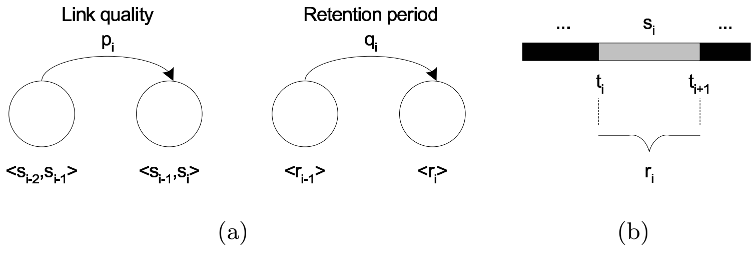


Fig. 3. (a) *Continued Move* mobility model and (b) *Smart Office* mobility model

The *Continued Move* model (Figure 3(a)) is based on a second-order Markov model for link states assuming that the next link state (s_i) depends only on the last two history states ($< s_{i-2}, s_{i-1} >$) and on a first-order Markov model for retention periods (that is, the duration a person remains at a certain link state area, r_i).[5] Four retention periods are considered, that are, (i) walking, (ii) short movement breaks, (iii) average working session, and (iv) stationary. The transition probabilities (p_i, q_i) can be extracted from real world traces.

The *Smart Office* model utilizes personal information, like a person's calendar entries for more accurate retention period and location estimation (Figure 3(b)). The model is based on a timeline metaphor describing locations (or link states s_i) which are entered at a particular point in time (t_i). The actual retention period is calculated as the interval between location changes ($r_i = t_{i+1} - t_i$).

[4] In link states EXCELLENT and MEDIUM, no specific failures are expected.
[5] Here, the location and the retention period are assumed to be independent.

3.3 Mobility-Aware Coordination Layer

The mobility-aware approach is designed as a modular layer termed *Mobility-Aware Coordination Layer (MobACL)* [12] based on shared DS middleware. Processes use the shared DS to coordinate themselves determined by the coordination pattern used, that is, they exchange data items by means of read and write operations, transactions, and notifications provided by the basic DS middleware. The MobACL uses an *internal state* to determine whether a process should access the shared DS (default mode) or whether the process should access a local copy DS instead (for example, in case the connection to the shared DS is lost). MobACL uses both current link state and retention period and the predicted values for the next link state and the next retention period (see Section 3.2) to derive this internal state.[6] In case the internal state changes, state transition operations, that are, copy, release of locks, and synchronize operations are invoked to maintain the consistency of the copy DS and shared DS.

Using the copy DS means that operations are tracked and postponed until the shared DS can be accessed again. MobACL uses *activity flags* for keeping track of each data item access. During the synchronization operation, these items are synchronized with the global space according to these activity flags. For example, if an item has been produced in the copy DS, this item is assigned by a *write* flag. During synchronization, this data item is produced in the shared DS. Hereby, the MobACL hides the distinction between shared and copy DS, and thus, the internal state, from the coordination pattern by providing an abstract API. A *logging function* allows to observe coordination activities by means of code instrumentation.

4 Prototypical Implementation

The MobACL reference implementation is based on three software systems and tools as shown in Figure 4. Based on MobACL distributed applications can be implemented by using the API specified. In the current implementation which includes four coordination patterns (Co-Patterns), MobACL assumes that the mobile device is capable of running an instance of the shared DS middleware CORSO [15].[7] MobACL is written in Java and accesses CORSO via the Java&Co API, which enables Java applications to access CORSO. Finally, MobACL makes use of the Orinoco WLAN client manager and ORINOCO classic gold PC cards which are compliant to the IEEE 802.11b standard for wireless link quality monitoring.

5 Experiments

In this section, the *coordination operation throughput* (also referred to as *throughput*), that is, the number of processed items of each of the four coordination

[6] In case no mobility model is configured, MobACL considers only the current link state to derive the internal state.

[7] In principle, it is also possible to implement a lightweight version of the copy space on devices not capable of running a middleware instance.

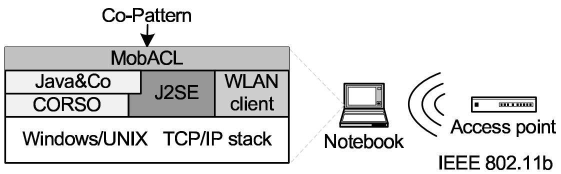


Fig. 4. MobACL reference implementation

patterns introduced in Section 2 is investigated under similar movement conditions. This number of processed items is used to argue whether the applied fault tolerance mechanisms can improve coordination in spite of mobile behavior.

Hereby, the following configuration cases are investigated. The *stationary* case is used as a reference for the coordination pattern's behavior under faultless conditions. The *without MobACL* case observes each coordination pattern on the move without application of fault tolerance mechanisms. Hence, mobility caused failures can be observed as decrease of throughput. When using the MobACL, the (*no prediction*) case refers to the use of MobACL which is solely reactive to the current link state. The two predictive cases are studied using the two different mobility models introduced in Section 3.2, that are, *Continued Move* and *Smart Office*. For the latter, a timeline similar to the movement is chosen which is used as an optimal predictor.

5.1 Experimental Setup

Figure 5(a) depicts the chosen movement schedule for all experiments. The motion covers continuous movement towards and heading away from a single WLAN access point and two discontinuous link changes at the end of the experiment.

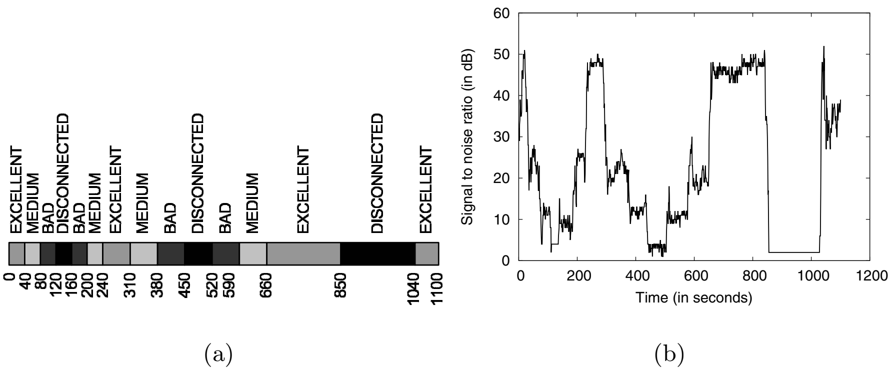


Fig. 5. (a) Time schedule of the experiment in seconds and (b) corresponding SNR measures used for all simulation-based experiments

All experiments have been carried out under the same conditions using a distributed simulator for synchronizing the start and the end of each experimental run between two hosts: (i) a stationary host and (ii) a mobile host. On the mobile host, the simulator is further used to simulate physical movement by replaying an SNR log file generated previously by when moving physically (Figure 5(b)). The simulator decreases the throughput in BAD and DISCONNECTED link states. While in link state DISCONNECTED throughput is decreased to 0, in link state BAD the throughput is decreased to 25 percent of the original throughput (this rate has been determined by iterative calibration). For all coordination patterns, processing of items is assumed to last one second.

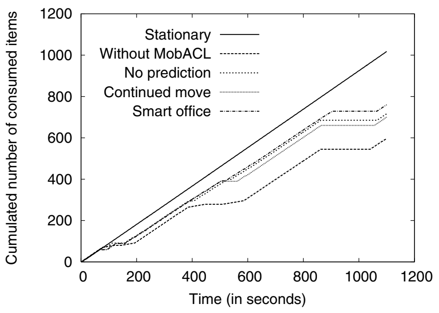
For each coordination pattern, two remote processes are involved. One process is assumed to execute on the mobile host and one on the stationary host. Due to space limitations, here a selection of four out of eight variants investigated is presented. However, in order to compare the results, for all processes executing on the mobile host, the process which requires input items from the other process has been selected for running on the mobile host. In detail, this is the *consumer* process in case of the *Producer/Consumer (PC)* pattern, the *subscriber* process of the *Publish/Subscriber (PS)* pattern, the client of the *Request/Answer (RA)* pattern, and an arbitrary peer in the *Mailbox (Mb)* coordination scenario.

5.2 Effects of Mobility and Fault Tolerance Mechanisms

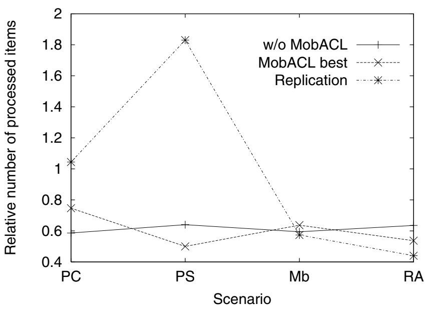
First, the example of the *Producer/Consumer (PC)* coordination pattern should explain and demonstrate the evaluation method. Figure 6(a) depicts the cumulated items consumed by the mobile consumer process over time. At the end of the experiment ($t = 1100$), the curve represents the sum of items consumed (processed) during the experimental run, which is 1018 for the *stationary* reference case. *Without MobACL*, this number decreases to 597, and the *smart office* mobility model, which is an optimal predictor, allows to raise this number to 760.

By following the curve *without MobACL*, the effects of mobility can be observed over time (for example, the disconnection at $450 \leq t < 520$). The discontinuous link state change at $t = 850$ can only be predicted accurately by the *Smart Office* mobility model. The *no prediction* MobACL configuration case is performing slightly better (716 total items consumed) than the *Continued Move* mobility model (700 total items consumed). This is due to prediction failures.

Similar curves have been generated for all coordination cases under investigation. In order to compare the results achieved for the differently coupled coordination patterns the throughput values (that are, the total items processed at the end of each experimental run) have been related to the reference *stationary* case. Hence, the *relative number of processed items* has been calculated for each pattern as depicted in Figure 6(b) and detailed in Table 1. The *MobACL best* case shows the best result achievable by MobACL, which is the *Smart Office* case for the PC pattern and the *no prediction* case for all other coordination patterns. For the two temporally coupled patterns (PS, RA), the current MobACL implementation worsens the results because it supposes to work on copies too



(a)



(b)

Fig. 6. (a) Mobility caused effects and achievements, which are detailed by (b) comparing different MobACL cases

Table 1. Comparison of mobility caused effects and achievements (relative number of processed items)

	Producer/ Consumer (PC)	**Publisher/ Subscriber (PS)**	**Mailbox (Mb)**	**Request/ Answer (RA)**
w/o MobACL	0.59	0.64	0.59	0.64
MobACL best	0.75	0.50	0.64	0.54
Replication	1.04	1.83	0.57	0.44

early. For the temporally uncoupled coordination patterns (PC, Mb), MobACL proved to be a potential means to improve throughput. While in case of PC the proactive fault tolerance mechanisms show clear beneficial results, the mailbox pattern test implementation covered the effects of increased availability of items to read by the writing part (postponed sending of messages) which is also supported by reactive mechanisms.

Replication of processes is another means to tolerate disconnecting coordination pattern participants. Hence, this technique has also been implemented by means of the shared DS and should be compared to the MobACL approach.[8] Here, the referentially uncoupled processes (PC, PS) can improve their overall throughput from a system's perspective (that is, the sum of items processed by all replicas), while the referentially coupled pattern cannot because the processes cannot be substituted (Mb, RA). Furthermore, adding new processes exhibits the potential for increased concurrency conflicts when accessing the shared DS.

6 Conclusion and Discussion

In order to evaluate the effects of movement caused failures in mobile computing scenarios such as collaborative emergency services, this work discussed

[8] The process executing on the mobile host was replicated on the stationary host.

four coordination patterns used by distributed applications. Based on the shared space-based computing paradigm the following patterns have been implemented: *Producer/Consumer*, *Publisher/Subscriber*, *Mailbox*, and *Request/Answer*. These patterns have been used as representatives of four different classes based on the temporal and referential domain. Weak wireless network conditions and disconnection periods have been considered as failure modes.

Reactive and proactive fault tolerance mechanisms have been introduced by the *Mobility-Aware Coordination Layer (MobACL)*. This layer observes the current wireless link state and makes use of predictors for future wireless link states and retention periods based on mobility models. As example mobility models, a second-order Markov model (*Continued Move)* and a rule based person centric model (*Smart Office*) have been used. The reference implementation of the MobACL is based on Java, CORSO, and the WLAN Orinoco client manager.

Experiments based on simulating movement have shown, that the decrease of throughput compared to the faultless situation can be compensated depending on the coupling characteristics exhibited by the coordination pattern. For the temporally uncoupled coordination patterns, the MobACL increased throughput from 60 percent (decrease caused by movement) up to 75 percent (*Producer/Consumer*). However, unnecessary state changes caused impairments of throughput for the temporally coupled patterns. Additionally, experiments have been carried out by replicating the process executing on the mobile device. For referentially uncoupled coordination patterns, that are, *Producer/Consumer* and *Publisher/Subscriber*, the total items processed in the system could be increased. However, replication is costly in terms of computing resources and may lead to increased concurrency conflicts when rising the number of processes.

To summarize, the reactive fault tolerance mechanisms of MobACL have improved coordination for temporally uncoupled processes. However, the success of proactive mechanisms depends on the accuracy of the mobility models. Hence, future work will concentrate mainly on the evaluation and introduction of new and accurate mobility models for prediction purpose.

References

1. Agere. Wireless LAN PC Card (Extended). Product Sheet, Agere Systems Inc., 555 Union Boulevard, Room 30L-15P-BA, Allentown, PA 18109-3286, USA, 2002.
2. P. Bellavista, A. Corradi, and C. Gianelli. Mobility Prediction for Mobile Agent-Based Service Continuity in the Wireless Internet. In *1st Int. Conference on Mobility Aware Technologies and Applications*, pages 1–12. Springer, 2004.
3. I. Burcea, H.A. Jacobsen, E. de Lara, V. Methusamy, and M. Petrovic. Disconnected Operation in Publish/Subscribe Middleware. In *2004 IEEE Int. Conference on Mobile Data Management*, page 39. IEEE Computer Society Press, 2004.
4. N. Busi, P. Ciancarini, R. Gorrieri, and G. Zavattaro. *Coordination of Internet Agents*, chapter 1: Coordination Models, pages 6–24. Springer, 2001.
5. G. Cabri, L. Leonardi, and F. Zambonelli. Mobile Agent Coordination Models for Internet Applications. *IEEE Computer*, 33(2):82–89, 2000.

6. L. Capra, W. Emmerich, and C. Mascolo. CARISMA: Context-Aware Reflective Middleware System for Mobile Applications. *IEEE Transactions on Software Engineering*, 29(10):929–945, 2003.
7. C. Cheng, R. Jain, and E. van den Berg. *Wireless Internet Handbook*, chapter 11: Location Prediction Algorithms for Mobile Wireless Systems, pages 245–263. CRC Press, 2003.
8. M. Cilia, L. Fiege, C. Haul, A. Zeidler, and A.P. Buchmann. Looking into the Past: Enhancing Mobile Publish/Subscribe Middleware. In *2nd Int. Workshop on Distributed Event-Based Systems*, pages 1–8. ACM Press, 2003.
9. S. Dustar, H. Gall, and M. Hauswirth. *Software-Architekturen.* Springer, 2003.
10. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns.* Addison-Wesley, 1995.
11. D. Gelernter and N. Carriero. Coordination Languages and Their Significance. *Communications of the ACM*, 35(2):97–107, 1992.
12. K.A. Hummel. Mobility-Aware Adaptation of Coordination Patterns. *Network and Information Systems*, 9(2):87–106, 2004.
13. T. Imielinski and B.R. Badrinath. Mobile Wireless Computing: Challenges in Data Management. *Communications of the ACM*, 37(10):18–28, 1994.
14. E. Kühn. Introduction: How to Approach the Virtual Shared Memory Paradigm. *Journal of Parallel and Distributed Computing Practices (online)*, 1(3), 1998.
15. E. Kühn. Coordination system. Patent EP0929864 B1, European Patent Office, D–80298 Munich, 2001.
16. E. Kühn and G. Nozicka. Post-Client/Server Coordination Tools. *Coordination Technology for Collaborative Applications*, 1364:231, 1998.
17. C. Mascolo, L. Capra, and W. Emmerich. Middleware for Mobile Computing (A Survey). In *LNCS*, volume 2497, pages 20–58. Springer, 2002.
18. C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich. XMIDDLE: A Data-Sharing Middleware for Mobile Computing. *Wireless Personal Communications*, 21(1):77–103, 2002.
19. A. Schill and S. Kümmel. Design and Implementation of a Support Platform for Distributed Mobile Computing. *Distributed Systems Engineering*, 2(3):128–141, 1995.
20. D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. *Pattern-Oriented Software Architecture – Volume 2.* John Wiley & Sons, 2004.
21. A.S. Tanenbaum and M. van Steen. *Distributed Systems - Principles and Paradigms.* Prentice Hall, 2002.

Application-Independent Session Mobility Between User Terminals

Priya Vijayakumar, Stefan Schmid, Simon Schütz, and Marcus Brunner

NEC Europe Ltd., Network Laboratories, Heidelberg, Germany
{vijayakumar, schmid, schuetz, brunner}@netlab.nec.de

Abstract. Today, users have more and more mobile and fixed devices with communication capabilities including phones, PDAs, Laptops, PCs, TV sets, HiFi equipment, etc. As the various devices have different capabilities, users want to be able to switch to the most suitable device depending on the current environment and context. In order to support service continuity, this implies that the moving of active sessions between terminal/devices must be supported. This paper proposes an application-independent approach for session mobility between different devices/terminals. We use the IETF Session Initiation Protocol (SIP) as basis and extend it to be used as a generic means for session mobility. We implemented and evaluated the system in a multi-media environment.

1 Introduction

The development of mobile communication systems is a prominent work item since people want to be connected wherever they go. Today's technology tries to keep pace with those mobility needs. The technologies stretch its applicability in all its directions possible to fulfill the needs of the user. Since users meanwhile own several devices, all of them with communication capabilities, the need of switching their sessions between devices is a relatively new user requirement in the mobile service area. Additionally, various other scenarios that involve external multimedia playback systems such as TVs, projectors, or audio systems, are becoming increasingly important.

The session mobility or user mobility as it is called in other contexts is known for quite some time. With session mobility in this paper we refer to the moving of an application session between two devices/terminals whereby the service is kept running. Session mobility can be interpreted in many ways. We assume session mobility is based around a user with different terminals or access to terminals. The session can run in one or more terminals at the same time. It depends on the user on how he would like to use the application in a multi-terminal environment. Note that we primarily focus on the same user moving the session, but do not exclude the moving of the session to a terminal of a different user. The latter however has a number of security implications, which we do not consider at the moment.

The user can choose either to have one of the sessions running on one of his devices or he can have the same application run on several devices. This kind of session mobility is proposed in variants for applications that use SIP as the signaling protocol (see related work below). In the SIP case, primarily voice call forwarding and third party call control are specified.

A key feature of our session mobility solution is that it is application-independent and does not rely on any specific communication protocol for the application sessions. In other words, its use is not limited to SIP applications, but can be used for any type of communication applications, such as RTSP based streaming applications. We merely assume that the applications/services do have a communication relationship with a content server, such as a web server, a streaming server, or a file server.

Note that in this work, mobility does not necessarily mean that the devices are operating in a wireless environment. The type of access network and end-to-end communication capability is not relevant for our work. The adaptation to different environments will be handled by the application or the network through media adaptation mechanisms that are outside the scope of this work. In order to illustrate the context of our work, we describe a small example scenario in the following section.

1.1 Example Scenario

Let us assume a user that is working on his personal device, say a PDA. The user is currently streaming some video content using the Internet. At some point, the user wants to move to another room (e.g. a conference room), where a large display, say a projection system, is available for the video playback, so that the other people in the room can also see the application session.

For this, the user needs to somehow "move" the application session to the projection system and be able to control what to show on the projector from his PDA. In case the user wants to show to application only on the projector, she/he can simply quit his PDA application after moving the latest session state to the projection system. Fig. 1 illustrates the simple scenario – a user has his PDA and then transfers the application session to a projection system in a conference room.

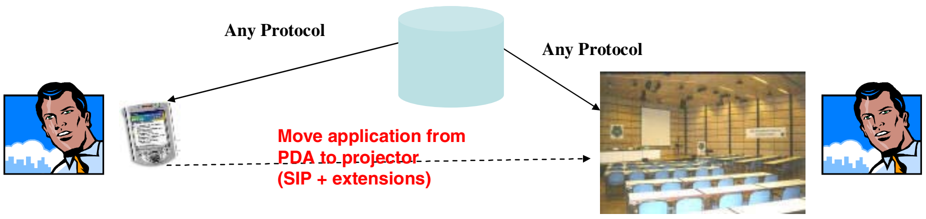


Fig. 1. Example Scenario Showing Mobility

Another scenario highlighting the flexibility of our solution could be a user surfing the Web on his terminal, who at some point wants to share the current content/view with a friend on another terminal. He/she simply needs to invite the friend at his terminal and give the necessary details to open up the Web browser and the current URL. In this case, the user wants to keep the application running on his/her own devices. He/she can continue browse locally and simply update the session state (e.g. the current URL), so that the other terminal also gets the same page displayed in the browser application.

1.2 Problem Statement

The main problem addressed by this paper is the move of an application session running on one device to another device. It is assumed that the content used by the application is accessible over a network. Furthermore, it is assumed that the content source (where the content is stored) does not keep state information of the session without sharing it with the client application. Good examples of such content are streaming media, web pages, and data files held on a server.

Several mobility approaches for application sessions need to be considered and differentiated. First, the user may want to move a session once and for all from the current terminal to the new terminal. Second, the user might want to have the applications running on both terminals at the same time, whereby the current terminal would be used to control the application session. In this case, synchronization of the session state between the controlling and the target terminal is required. For example, in the case of video streaming, when the user presses or releases the fast forward button, this action and the resulting session state must be synchronized between the terminals.

The user must also be able to close the application on one device any time but keep the session between the application and the content server running on the second device.

A key requirement for any session mobility solution is to be application-independent so that the same mechanism can be used for a wide range of different applications that users may want to move between terminals. Such applications typically maintain only small states on the local terminal, which can be transferred without any complication. As a result, the session mobility agent must be able to handle general state information in order to transfer any application session.

Finally, the location of remote terminals with appropriate capabilities has been identified as an open problem for some usage scenarios. However, since this problem is orthogonal to the problem of moving a session between terminals, it is considered outside the scope of this paper. Various solutions for this problem can be integrated into the session mobility approach, where one specific solution is explicitly supported, namely the use of a SIP proxy, which handles the location of devices with certain capabilities.

2 Application-Independent Session Mobility Architecture

This section shows an overview of the proposed architecture. We decided to use the Session Initiation Protocol (SIP) protocol as a basis for the communication between the terminals, where sessions are moved. It is important to note here that we do not assume that the applications themselves use SIP for the application session control.

A modified SIP user agent, called Session Mobility Agent (SMA), on both terminals involved in the session handoff is used to control the transfer of an application session. The SMA on the source terminal is responsible to initiate the mobility procedure and to control the transfer of the session state. For this, the SMA must be able to acquire the session state from the user application. As a result, the application must support the SMA interface in order to provide the internal session state to the SMA. The SMA must be able to marshal the state information it got from the user application and send it to the target terminal using the SIP protocol.

It is important to note here that the user applications on the different terminals do not have to be identical. It is fully sufficient that they support equivalent functionality (for example, media players from different manufacturers that are able to receive and play back mp3 files using the same signaling and streaming protocols). To achieve this, the user applications must implement the session mobility application interface exposed by the SMA.

The SMA doesn't need to be aware of the capabilities of the user application. It must merely know that the application exists and how to bootstrap the application (in case the application is not already running). This reduces the complexity of the SMA trying to know or understand the syntax or semantics of the session state provided by the user application. The SMA simply encapsulates this information and passes it to the target application, which is expected to understand it and handle accordingly. The SMA also doesn't need to be concerned about the other features of the user application.

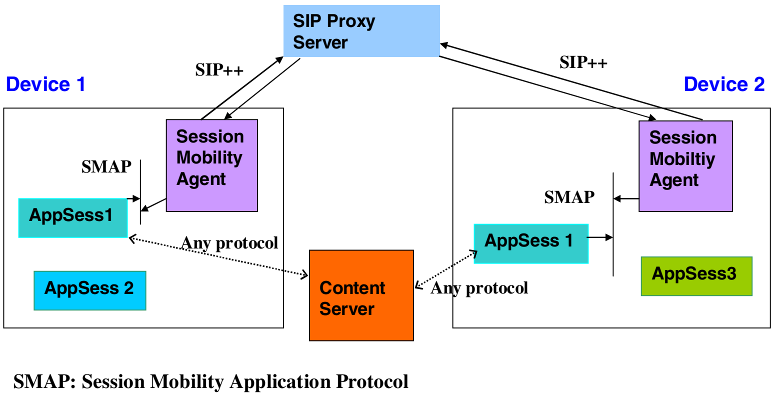


Fig. 2. Session Mobility Architecture

The SMA of the source terminal must know some parameters of the other peer prior to the start of the signaling between the two. First and foremost it must know how it can reach the destination terminal. Both terminals could be present inside the same network or they may be reachable though another network. So there are different possible ways of doing that.

Our architecture proposed the use of a SIP proxy, which can handle the routing of session mobility requests to appropriate terminals/devices and also serve as a capability broker. For example, the SIP proxy can resolve SIP URLs in a contextual manner – for instance, the URL "projector@localnetwork" can be resolved to a local device that has been registered as a projector. Additional capability information can be held by the SIP proxy in order to help roaming terminals visiting a foreign network to find appropriate devices to share or move a session. To make this work, the SMA will register a terminal along with its capabilities with the SIP proxy. A simple way is to encode a device's capabilities or functions directly into the SIP address of the terminal (e.g. mp3_player@localnetwork). However, this simplified approach is only useful for the registration of the very basic functions or capabilities of a terminal.

In case the target terminal is not known to the terminal maintaining the session application in the first place, and no SIP proxy is available for the re-routing of the application session, the SMA on the original terminal can use the Service Location Protocol (SLP) [17] in order to locate and the potential target devices. SLP allows for directory based search of devices with a certain capability or location.

After the SMA on the target device has started the desired user application, the SMA must also give back the response to the originating SMA, whether the application was successfully started or not. After moving the application from one terminal to another, the target terminal must also be able to receive updates from the old device concerning state changes of the application session.

As illustrated in Fig. 2, all potential session mobility terminals must have the SMA installed. Each has got a set of applications, which are interfacing with the SMA. For example, AppSess1, AppSess2, AppSess3 are applications that are running on the terminal. The applications communicate with the SMA using the so-called Session Mobility Application Protocol (SMAP).

The proposed session mobility approach supports two basic modes for moving a session, namely the *push mode* and the *sync mode*. In the *push mode*, the SMA sets up a mobility session to another SMA. The target SMA then starts the same or an equivalent application on the target device. The sending SMA provides the session state of the moved application as part of the INVITE message. Once this message has been successfully received by the SMA on the target terminal, it will end the signaling session upon which the original application will be terminated.

In the *sync mode*, the sending SMA also establishes a SIP signaling session with the receiving SMA. The SMA at the receiving end not only starts the application as specified in the INVITE, but also subscribes to the events responsible for synchronizing both application sessions. This leads to the notification of events in case there are some synchronization steps issued by the user. In this case, the application session is forked and executed on both the original as well as the target terminal. The event-based synchronization mechanism ensures that both application sessions are kept in sync.

2.1 Session Mobility Application Protocol (SMAP)

The information exchange between the application and the SMA process on the local terminal can be achieved by any inter-process communication means. Note that the best choice might be operating system specific.

The semantic of the information exchange follows the message flow illustrated in Fig. 3. The meaning of the different messages is further described here:

MOVE_SESSION: The SMA sends this request message to the user application indicating that it wants to move the current session to another terminal. This message triggers the user application to provide the current session state to the SMA. This message has typically no parameters, as it only indicates the start of the session mobility procedure. In response to this message, the user application provides the session state of the current communication session as part of the *MOVE_SESSION_READY* message. This contains the basic session state information, such as communication server, the media content, the streaming protocol, etc.

GET_SESSION_DATA: This request message is sent from the SMA to the application to get an update of the current state of the application session. This for example includes the current timestamp of a streaming session. The application passes whatever state information is necessary to synchronize the remote session, which of course depends on the type of application/session and the protocol state machine used for accessing the content. The application responds to this message with the *GET_SESSION_DATA_READY* message, which carries the required session state information.

SET_SESSION_DATA and SYNC_SESSION: These messages are used by the SMA to signal the initial or up-to-date session state of the source application to the target application. The corresponding response messages indicate whether or not the state update was successful or not.

APPLICATION_SHUTDOWN: This message is used by the SMA to signal the original application to halt after the application session has been successfully moved to the target terminal.

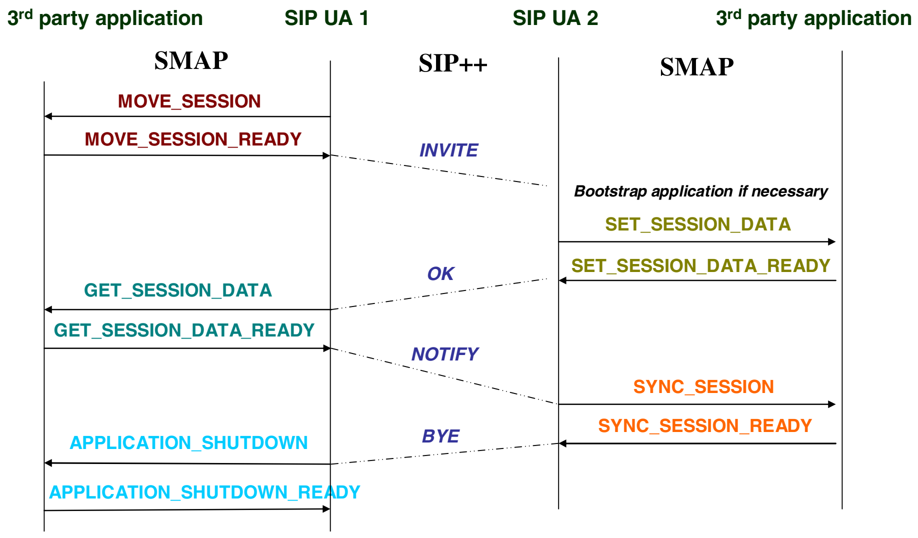


Fig. 3. Communication between 3rd Party Application and SIP UA

2.2 SDP Extensions for Session Mobility

The SIP-based session mobility protocol requires only two minor extensions to the Session Description Protocol (SDP), which is normally used by SIP to describe the session parameters/settings during session setup. First, we introduce a new Event Package for synchronization purposes, and second, we use SDP also to carry application specific session state between terminals.

The event package uses the SIP *SUBSCRIBE* and *NOTIFY* messages to transport event related information. The event specification must give details about the type of application and also about the parameters used by the application in order to communicate the state changes to the peer on the other end. We named the event package *Session-Sync***.**

The *SUBSCRIBE* message does not contain any body. It merely specifies the time in seconds for which the event subscription is valid. Typically, the subscription is started as soon as the SIP session mobility is setup. Every time a subscription is renewed the sequence counter *Cseq* is increased in order to differentiate between the old subscribe message and the new subscribe message. Otherwise, the stack will be neglecting the new subscribe messages thinking that they were being retransmitted.

The *NOTIFY* messages use the event package format for conveying the session state information. They have a field called *Event*, which specifies what event the message is reporting. The *Content-Type* tells the format of the report (we currently support SDP). This message is sent every time the SIP UA triggers the *Session-Sync* event.

It should be noted here that synchronization event package should not be used for very frequent synchronizations of applications (e.g., in the order of milliseconds) as this can cause a heavy burden on the SIP stack due to the frequent notifications.

The second change required by our session mobility protocol is related to the exchange of application specific session state information. Unlike in normal SIP scenarios, where session information are only exchanged at the beginning of a SIP session, we use SDP also for the synchronization procedure while the SIP session is established in order to update the application sessions on the different terminals.

In order to allow the transport of application specific session state, we also extended the normal SDP body. As the actual session state information differs from application to application, we introduced some generic fields that allow the applications to transfer their session state in an opaque manner. For example, a video streaming application based on RTSP [12] may use these fields to pass the URL of the video server and file, as well as the timestamp of the current position of the video playback to the target application.

3 Evaluation

This section evaluates the proposed session mobility scheme qualitatively and quantitatively. The performance evaluation is based on experiments carried out with our prototype implementation, which has been founded on the Vovida SIP Stack (www.vovida.org) and tools, the Darwin Streaming Server (www.apple.com), and the gmp4player (www.mpeg4ip.net) as client application. Our implementation of the Session Mobility Agent (SMA) is based on the Vovida SIP stack, where we have integrated the required protocol extensions (e.g. the *SessionSync* event package) and the Session Mobility Application Protocol (SMAP). The gmp4player application has been extended to interwork with the SMA using the SMAP protocol.

Applications that maintain small amount of state information for their sessions are ideal candidates for mobile application. Examples include streaming applications, where the URL of the streaming file and the playback timestamp are the only state required to move a session, and presentation applications where the presentation data/file is retrieved from a content server; in this case, the required session state would consist of the content server and the file name as well as the current slide number.

In order to support session mobility between applications that are not identical, but support equivalent functionality, it is necessary to classify and group compatible applications. It is therefore suggested to define common application classes (e.g. RTSP streaming applications or Web browsers), which allow transfer of application sessions among applications of the same class. A standardized way of transporting the session information is then also needed to achieve interoperability between different applications that support the same functionality.

Our implementation of the SMA also supports a GUI, which allows interactive authorization of session mobility requests by the user of the terminal. In case of a session mobility request, the SMA GUI will display a pop-up message with the task to either accept or reject the session mobility request. In case this feature is activated, the session mobility delay will also depend on the responsiveness of the end user.

The SMA must be able to address the applications in order to communicate with them via the SMAP. This is configurable in the Session Mobility agent configuration file. Enhancements could be done so that the agent is able to recognize or automatically learn a list of applications supporting session mobility and currently running on the device.

Another observation is that the notification events (*SessionSync*) should not be triggered too frequently. The reason is that the SIP protocol is not designed to carry a high number of messages in a small amount of time. For example, notifications could be sent when the user moves the slider of the player, while browsing through the movie. When every small move would triggers a state update, it could overload the network as well the other end system, since the state change involved may also imply an interaction with the content server. As a result of this, the notification events should only be raised when necessary (i.e., in the event of important/significant actions). In case periodic notifications are used (e.g., to automatically synchronize the presented slide or Web page on the presentation device), the notification interval should not be decreased to less than 1-2 seconds.

Security in the context of session mobility is a rather huge topic and would require a very detailed analysis on its own. Security can be mainly divided into two categories. The first is to setup a secure SIP call, where some work is done, which is in the process of being standardized. Second, it is to get the SIP call passed through security devices like NATs and firewalls. A detailed analysis of this is out of scope for this paper and readers are referred to solutions described in [15] and [16].

In the following we present also preliminary performance results of our prototype SMA implementation when used in conjunction with our session mobility enabled video streaming application. The scenario outlined in Figure 1 is used as a test case for our measurements. The experiment was carried out in a dedicated test network, which consisted of three PC systems (Pentium IV, 1.7 GHz, 256 MB RAM).

The main target of our experiments was to determine the session handoff time in the above environment. The overall time to move the session is calculated based on the time difference from sending the initial *INVITE* message for the session handoff request until the session move has been acknowledged. This handoff time includes all delays that are associated with the initial session mobility procedure. I.e., when the SMA on the target system receives the *INVITE* message, it first has to start the respective application, and then if successful, it needs to update the session state of the newly started application using the SMAP. Only then, the SMA is able to respond to the *INVITE* message. Our findings show that the time for the initial session mobility procedure is around 1.7 seconds (with 0.1 seconds standard deviation over 20 runs). Approximately 1 second of this delay is required just for starting up the gmp4player at the target terminal as it includes the initiation of the streaming session with the content server.

Moreover, the analysis of the session mobility delay has revealed the following results. First, depending on the distance and connectivity between the original terminal and the target terminal, the session mobility delay can vary significantly. Second, depending on the external user application, the time required to instantiate the application on the remote terminal varies. Third, applications also introduce extra delay when they have to initiate communication with remote entities (e.g., a streaming server on the network) as part of the session setup that may vary significantly depending on the distance and connectivity to the remote entity. Fourth, in case the SMA requires authentication, the additional message exchange will also introduce extra delay to the initial SIP handshake.

We think this time scale is completely tolerable for the application of session mobility in the context of streaming applications. It is important to note here that the user has to wait this time only for the startup of the session on the target terminal. The successive synchronization procedure (which immediately follows the initial handoff procedure) ensures that new session runs in sync with the original.

4 Related Work

There are a number of SIP mobility solutions for session mobility when the application itself uses SIP as the session signaling protocol between the application and the server [5]. In our approach we are more generic, because it works for all application independent of the protocol used at application level. The *SIP third party call control* (*3pcc*) belongs into this category [4]. It is an entity that can set up, manage sessions between two or more participants. It is responsible for inviting other participants into the session and it controls the session when some participants leave. It is mainly used in conferencing services. Many of the PSTN services are achieved through *3pcc* like call transfer from a secretary to the boss. One of the major applications where it is awaited in use is click-to-dial or push-to-talk. With a press of one button, the session between the user and the person at the other end can be achieved.

Another mechanism of the same category is the SIP *REFER* method [6], upon which the recipient should contact a third party using the contact information provided by the *REFER* method. It contains the URL of the third party to be contacted. The third party must accept the *REFER* message only if it recognizes the URL and is able to make further processing of the session [10].

Moreover, we are aware of several other approaches to session mobility. In the following we just show a small subset of the work available.

The *Mobile People Project* [13] utilizes Personal Proxies to route communications to a mobile user, independently of the user's location and applications she/he is currently using. This is more a network level change, than moving sessions on end-systems, since it include also routing and transcoding issues, which are needed, but not the focus of this paper.

The *browser session preservation and migration* work [14] provides centralized personal data storage, specifically for browser session state preservation and migration, rather than a broader range of applications. The way of storing the browser session could be used in our approach when moving browser sessions form one device to another.

5 Conclusion

This paper proposes a solution to achieve session mobility between user terminals in a generic way. Our solution is generic regarding the application session that can be transferred as well as the protocols that are used by the applications. The method includes a simple combination of SIP/SDP messages that indicate the mobility of a session. The messages include information about the application (e.g. name and/or class of the application) and the current state of the application session to be moved.

For the session mobility scheme to acquire the current session state, the Session Mobility Agent needs to communicate with the application. A key feature of the proposed session mobility scheme is its ability to handle arbitrary applications. An application-independent Session Mobility Application Protocol has been developed to enables session state exchange between the Session Mobility Agents and arbitrary applications.

Additionally, a new SIP event package, called *SessionSync*, has been created in order to indicate state changes of application sessions that have been moved and need to be kept in sync with the original session. Finally, a basic experimental user scenario has been set up in order to test and evaluate the session mobility scheme. The evaluation has shown that the proposed scheme works well for streaming application.

As part of future work, we plan to develop a discovery mechanism for Session Mobility Agents in order to facilitate automated session transfers between terminals. We plan to deploy a SIP registrar that stores the capabilities of individual terminals. For example, a device that has projection capabilities could register itself as a projection system. Later, when a user or application looks for an appropriate device/functionality for a particular application session, the SIP registrar can be queried to identify a suitable terminal. Moreover, we also expect to enhance our *SessionSync* event package in order to support additional parameters that will be required when new scenarios are considered. In particular, we plan to study multi-user, multi-terminal session mobility scenarios in the future.

Acknowledgements

The authors are partly funded by *Ambient Networks*, a research project supported by the European Commission under its *Sixth Framework Program*. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the *Ambient Networks* project or the European Commission.

References

[1] M. Handley, H. Schulzrinne, E. Schooler, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
[2] M. Handley, V. Jacobson, "Session Description Protocol", RFC 2327, April 1998.
[3] Popescu, “Supporting Multimedia Session Mobility using SIP”, CNSR 2003 Conference, Moncton, New Brunswick, Canada, May 15-16, 2003.
[4] J. Rosenberg, J. Peterson, H. Schulzrinne, G. Camarillo, “Best current practices for Third Party Call Control in the Session Initiation Protocol”, draft-ietf-sipping-3pcc-05, SIPPING Internet Draft, October 27, 2003.
[5] H. Schulzrinne, E. Wedlund, "Application-Layer Mobility Using SIP", ACM Mobile Computing and Communications Review, Vol.4, No.3, July 2000.
[6] R. Sparks, "SIP Call Control Transfer", Internet Draft, draft-ietf-sip-cc-transfer-05, July 18, 2001 (expired).
[7] J. Glasmann, W. Kellerer, H. Müller, “Service architectures in H.323 and SIP: A Comparison”, IEEE Communication Surveys, Vol 5(2)., 2003.
[8] B. Roach, J. Rosenberg, B. Campbell, “A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists”, Internet Draft, draft-ietf-simple-event-list-04, June 13, 2003.
[9] B. Roach, “Session Initiation Protocol (SIP)-Specific Event Notification”, RFC 3265, June 2002.
[10] R. Sparks, “The Session Initiation Protocol (SIP) Refer Method”, RFC 3515, April 2003.
[11] Roach, "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.
[12] H. Schulzrinne, R. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.
[13] B. Raman, R.H. Katz, and A.D. Joseph, "Universal Inbox: Providing Extensible Personal Mobility and Service Mobility in an Integrated Communication Network", Workshop on Mobile Computing Systems and Applications (WMSCA'00), December 2000.
[14] H. Song, H. Chu, S. Kurakake, “Browser Session Preservation and Migration”, WWW’2002, Hawaii, May 2002.
[15] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, A. Rayhan, “Middlebox Communication Architecture and Framework”, RFC 3303, August 2002.
[16] C. Martin, A. Johnston, “SIP Through NAT Enabled Firewall Call Flows”, draft-martin-midcom-sip-natfw-callflows-00.txt, Internet Draft, February 2001.
[17] E. Gutman, C. Perkins, J. Veizades and M. Day, “Service Location Protocol, Version 2”, RFC 2608, June 1999.

Using Multiple Communication Channels in a Mobile Agent Platform

Theodoros E. Athanaileas, Ioannis E. Foukarakis, Dimitra I. Kaklamani, and Iakovos S. Venieris

School of Electrical and Computer Engineering,
National Technical University of Athens
Fax:+30-210-7722291, Tel:+30-210-7722287
ifouk@esd.ece.ntua.gr

Abstract. This paper describes a flexible architecture that enables the adoption of additional communication channels in a mobile agent platform. The architecture extends an already implemented Web Service [1] based mobile agent platform by replacing communication mechanisms with more advanced components, and without modifying the platform's core components. The proposed architecture uses well-known design patterns to better define its distinguishing parts. Sockets are used as the first alternate to SOAP communication. Finally, the two different communication channels are compared and benchmarked.

1 Introduction

During the last years, research on mobile agent platforms has focused on porting existing platforms or developing new platforms for different environments such as mobile devices [3], web servers or even the Grid. These platforms use common communication channels [2], [4], usually dictated by limitations of the runtime environment (i.e. HTTP for mobile devices [6]). Although architectures and agent communication languages [5] have been proposed for enabling interaction with other agent platforms, most platforms rarely use more than one communication channel. This limits the mobile agent technologies, because of the lack of interoperability between platforms running on top of different environments.

In order to bypass this problem, we propose a modular architecture that enables the adoption of multiple communication protocols. The existence of these mechanisms is transparent to the core platform, thus minimizing the cost of migrating existing mobile agent applications to the new architecture.

2 The Core Agent Platform

The platform described in this paper is based on an already existing platform implemented in Java which makes use of Web Services [7]. Web services are collections of operations covered in modular, self-contained applications that are accessible over a network and generally the Internet, through standardized XML messaging. Communication between platform components is achieved by using SOAP (Simple Object Access Protocol) and specifically SOAP-RPC. With SOAP-RPC, an object situated on a remote host invokes a method and gets the result when the method returns.

The platform has been designed according to the OMG MASIF [8] specification. The concepts of places, agencies and regions used by other existing platforms have been adopted.

The region is used for providing directory services. Upon creation, movement, end of execution and other major events in an agent's life cycle, the object is notified by the agencies that are registered to it, so that it keeps track of all the changes. What is more, an agency or an agent (through its agency) can send queries to the region in order to retrieve information about other agents or other agencies.

The main components of the platform are the agencies. Agencies provide the agents with an environment to run and a communication gateway to other agencies and/or agents. Agency objects provide the actual runtime environment for both Stationary and Mobile agents. Their main operations are to create, store, move agents and to allow them to communicate with other platform components.

Each agency is divided into places. The concept of places is used in order to provide grouping capabilities for the agents. A place groups the functionality within an agency, encapsulating certain capabilities and restrictions for visiting agents helping this way the administrator and the programmer to separate different agents.

A major part of the platform is the workers, who are responsible for performing specific tasks, such as requesting the creation of a new place in an agency or registering an agency to a region. There are two categories of workers: those used for agency-to-agency communication and those used for agency-to-region and vise versa communication. The Workers API acts as a middle-tier between the platform and the communication channel, thus making it possible for another protocol to be easily embedded into the platform communication mechanism.

3 Sockets

It is known that a socket is one endpoint of a two-way communication link between two programs running on a network. A socket is bound to a port number so that the communication protocol can identify the application that data is destined to be sent. There are two protocols that can be used for socket programming: UDP [9] (datagram communication) and TCP [10] (stream communication). UDP is an unreliable protocol – there is no guarantee that datagrams sent will be received in the same order by the receiving socket, whereas TCP is a reliable protocol – the packets sent are guaranteed to be received by the receiving socket at the same order in which they were sent. It is more than obvious that for an agent platform – where agents must be reliably transferred – only the use of TCP sockets is acceptable.

4 Sockets vs. Soap

Apart from the advantages and comforts that SOAP and Web Services offer in designing distributed applications, there are also some drawbacks that may cause some problems in certain situations.

First of all, Web Services are based on the HTTP protocol, thus making use of port 80. But this port is also used by any other program which uses the HTTP protocol. This means that security precautions can become complex. It must be noted that SOAP does not make the HTTP protocol less secure than it has always been, but it makes use of it considerably more sophisticatedly than usually and therefore a more sophisticated security model than that usually used for HTTP traffic is demanded.

By using plain sockets, security can become a less complex problem. Each platform component can use any non well-known port, which means that it does not have to share its port with any other application, thus making security controls much easier. What is more, the use of Secure Socket Layer (SSL) can be easily imported.

The most important disadvantage of using SOAP is lack of performance due to the requirement to parse and transport XML. The use of XML places a heavy burden on the system. Despite the relative simplicity of XML, there are huge memory and CPU requirements compared to other solutions. Generating and parsing XML documents is a time consuming task that also needs a lot of memory compared to the actual data that is in those documents. It must be noted that these documents have to travel across the network. With XML the actual information is only a small part of the total data, which results in a high overhead and wasted network bandwidth.

By using plain sockets, we achieve a performance gain. Sockets are flexible and give the ability to implement an application so that only absolutely necessary information is sent over the network, resulting in both higher communication speed and bandwidth efficiency. In order to implement a distributed application, a message-passing system must be adopted. The messages passed can contain only the necessary data, thus contributing to the efficiency of the application. On the other hand, applications based on SOAP have to transfer whole web pages for each request.

The trade-off of using sockets to implement a distributed application such as a mobile agent platform is that both the server-side and the client-side have to provide mechanisms to make the data transferred in the messages useful. This makes the application more sophisticated.

5 Socket Agent Platform Architecture

The architecture of the socket platform is depicted in Figure 1.

The communication between the components of the Socket Platform is implemented through a message-passing approach. Therefore, we make use of a RequestMessage class which contains all the information needed for the communication. This class has fields that determine the method to be invoked on the target component as well as the method arguments. A ReplyMessage class is used to store the result of the method. An instance of the first class is sent by the client to the server in any communication attempt, while the server sends an instance of the ReplyMessage class to the client when the communication request has been processed. This transaction is made possible by using the Java Serialization feature and object input and output streams.

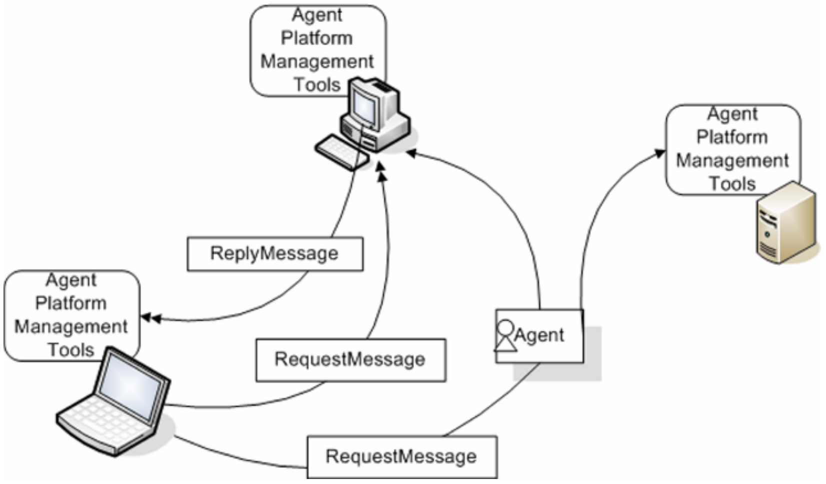

Agent Platform Management Tools
Agent Platform Management Tools
Agent Platform Management Tools
ReplyMessage
RequestMessage
RequestMessage
Agent

Fig. 1. The general architecture of the Socket agent platform

5.1 The Server Side

In order to achieve an operative platform using sockets, we have made use of a classic multi-threaded socket server, which contains an instance of the Agency or Region class. The server creates a ServerSocket and waits for communication requests. Whenever a communication request is attempted, a new thread is created which reads a serialized object of the RequestMessage class through the socket input stream. Depending on the type of the message, the server chooses the appropriate method of the component to be invoked and the result returned by the method is stored in an instance of the ReplyMessage class, which in turn is sent back to the client. While in a SOAP Platform every component is identified by a URL, in a Socket Platform the identification of a server needs the host address the platform component is running on and the port number the server is attached to.

5.2 The Client Side

The client side of the Socket Platform is implemented through the Workers API. The Workers API manages to maximize code reusability, but most importantly it offers a transparent way of communication independently of the protocol used by the platform. This means that using the Workers API along with the add-on of the WorkerFactory discussed later, it is possible to achieve communication with any platform using either SOAP or sockets, or even any other future-supported protocol (such as RMI).

The workers are the objects which create the socket connection to the remote host. Whenever there is a need for interaction with a remote platform component, a worker is created. The worker creates the appropriate instance of the RequestMessage class, forwards the message to the server and blocks until a ReplyMessage object is received from the socket input stream. In order to make a connection attempt, the worker only needs the address of the host the Agency or Region is running on and the number of the socket the server is attached to. These are the two arguments demanded by any worker, along with other method-specific arguments that some workers may need.

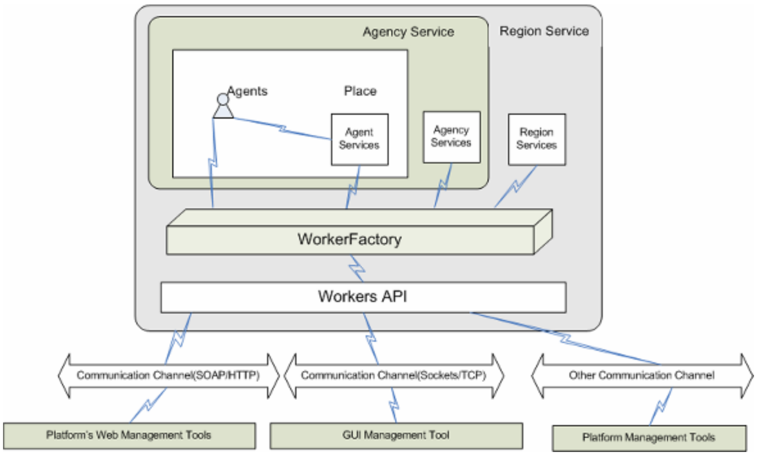

Agency Service
Region Service
Agents
Place
Agent Services
Agency Services
Region Services
WorkerFactory
Workers API
Communication Channel(SOAP/HTTP)
Communication Channel(Sockets/TCP)
Other Communication Channel
Platform's Web Management Tools
GUI Management Tool
Platform Management Tools

Fig. 2. The architecture of the protocol-independent agent platform

6 Protocol-Independent Platform

One of the main objectives of our work was to achieve complete transparency at the interaction between components of the platforms, independently of the platform protocols. In other words, whether the communication attempt is made towards a SOAP platform or a Socket platform or a platform based on any other protocol, the approach should be the same. The architecture of this protocol-independent platform is depicted in Figure 2.

In order to achieve this protocol independency, first of all we introduced to the components of the platform an Identifier class which stores information about the protocol as well as the other objects which construct the identification needed to communicate with the component according to each protocol (for example the host address and the port for the Socket platform).

The most critical step towards the direction of protocol-independency was finding a way to dynamically choose the right kind of workers for any supported protocol. This means that whenever communication is needed, the component, which possesses only the target Identifier and the arguments for the method to be called in the remote platform, should be able to get the right worker by passing this information to an object. This object is the WorkerFactory discussed in the next paragraph.

6.1 The WorkerFactory Class

The WorkerFactory diagram is depicted in Figure 3.

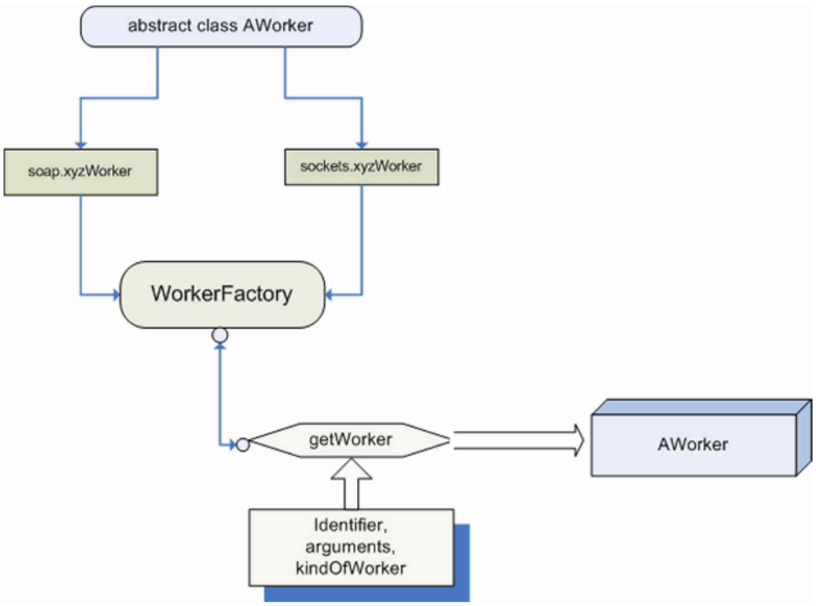


Fig. 3. The WorkerFactory pattern

In this figure, AWorker is the abstract class that all soap workers and socket workers are derived from. The AWorker class implements or defines fields and methods which are common to all worker classes. Every worker class needs to extend the AWorker class. The WorkerFactory is a class that decides which of these subclasses to return depending on the arguments given. It contains a getWorker method which passes some arguments to identify the address as well as the protocol of the target, the arguments of the method to be invoked at the target and the kind of worker to be returned. The method returns the appropriate instance of the class AWorker. It must be noted that which instance is returned does not matter to the programmer since they all have the same methods, but different implementations. The decision about the proper instance of the class is completely up to the WorkerFactory and is made according to the protocol of the target and the kind of worker needed. The other arguments that are passed to the getWorker method are needed only for the instantiation of the worker.

7 SOAP Platform and Socket Platform Comparison

In order to benchmark the usage of the two different communication channels, a simple agent was developed. This agent moves between two agencies hosted on different computers. Upon creation, the agent creates a byte array that is used as a payload, migrates to the remote agency and returns. The byte array acts as a way to control the agent's size in memory. The round trip time is used as a metric for the platform performance. This scenario is repeated several times for various payload sizes and average round trip time is taken into account for each payload size. The results from this benchmark can be seen in Figure 4.

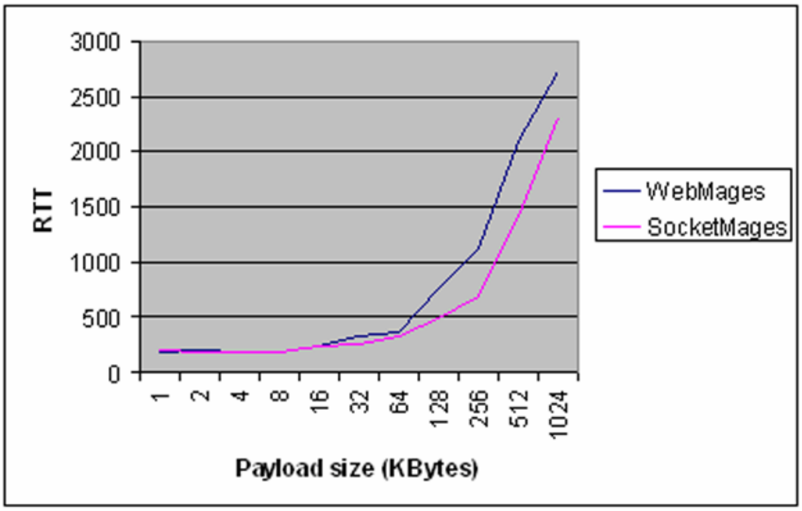


Fig. 4. Socket and SOAP comparison

For small payload sizes (1-64 Kbytes), the average round trip time for the two different workers API's is similar. But for larger payload sizes, sockets are faster than SOAP. As it has been noted earlier, this is expected because of the time required to parse larger XML documents.

8 Conclusions and Future Work

In this paper a multi-protocol mobile agent architecture was presented. The current architecture defines endpoints where handlers for different communication channels can be attached, further extending this way the platform. The main advantage of this architecture is that it doesn't require changes in the core of the platform. Tests run using SOAP and sockets as the communication channel have shown that sockets offer better performance.

Our current efforts focus on extending the platform by implementing additional Worker classes conforming to various protocols such as CORBA and RMI.

References

1. W3C consortium, "The Simple Object Access Protocol", http://www.w3.org/TR/SOAP/
2. JADE mobile agent platform, http://jade.tilab.com
3. J. Cao, D.C.K. Tse, and A. T.S. Chan, "PDAgent: A Platform for Developing and Deploying Mobile Agent-Enabled Applications for Wireless Devices", in Proc. 2004 International Conference on Parallel Processing (ICPP'04), August 15 - 18, 2004, Montreal, Quebec, Canada, pp. 510-517

4. M. Aleksy, A. Korthaus and Martin Schader, "CARLA – A CORBA-based Architecture for Lightweight Agents", in Proc. IEEE/WIC International Conference on Intelligent Agent Technology, October 13 - 17, 2003, Halifax, Canada, p. 111.
5. FIPA 97 Part 2 Version 2.0: Agent Communication Language Specification, http://www.fipa.org/specs/fipa00003
6. LEAP Lightweight Extensible Agent Platform, http://leap.crm-paris.com/
7. I. E. Foukarakis, A. I. Kostaridis, C. G. Biniaris, D. I. Kaklamani and I. S. Venieris, "Implementation of a Mobile Agent Platform Based on Web Services", Proceedings of the 5th International Workshop on Mobile Agents for Telecommunication Applications (MATA'03), Marrakech, Morocco, October 8–10, 2003, Lecture Notes in Computer Science, Vol. 2881 / 2003, Springer–Verlag, (ISBN 3–540–20298–6), pp. 190–199.
8. OMG MASIF, "Mobile Agent System Interoperability Facility (MASIF) specification", ftp://ftp.omn.org/pub/docs/orbos/97-10-05.pdf
9. RFC 768, User Datagram Protocol, http://www.rfc-editor.org/rfc/rfc768.txt
10. RFC 793, Transmission Control Protocol, http://www.rfc-editor.org/rfc/rfc793.txt

Challenges in Modelling and Using Quality of Context (QoC)

Michael Krause and Iris Hochstatter

Mobile and Distributed Systems Group,
Ludwig-Maximilian-University Munich, Germany
{michael.krause, iris.hochstatter}@ifi.lmu.de

Abstract. Context-awareness in the vision of Ubiquitous Computing implies more than just the integration of sensor data into applications. It promises vast benefits from automated sharing and combining of knowledge about current situations. We emphasize the necessity of quality of context (QoC) to enable and improve the automatic rating and processing of context information. We derive requirements for handling and modelling of QoC by analyzing the context provisioning process and show that existing integrations of QoC into context models are not adequate. And we concretize our understanding of QoC by distinguishing between information-inherent QoC and application-dependent considerations.

1 Introduction

In a highly dynamic environment with many mobile entities, where adaptive systems try to find context-information at run-time, the urgent need arises to evaluate context information that is gathered from outside the system domain. We want to give a deeper insight into the demands of handling and modelling quality of context and show that these demands exceed existing modelling approaches. This paper is structured as follows: at first we show the necessity of QoC by summarizing the discussion on this topic, then we analyze a general context provisioning process and derive requirements for QoC-handling that are more complex than those recent research work did take into account up to now. This we show in a section of related work.

2 The Necessity of QoC

Ubiquitous Computing takes its advantages of a rich-sensor-environment and the sharing and combining of knowledge, so that the use of context information allows to support pro-actively the interaction between human and machine or between machines. But context information is subject to many error sources:

– Necessary context information (or context sources) might be not available.
– Context information might be out-dated and no longer applicable to the current situation.

- Physical constraints and temporary effects limit the precision of sensors.
- Default profile information might not apply to the current situation.
- Reasoning rules which often base on probabilities do not apply in certain situations and they could therefore derive wrong context information (Not every detected noise means that a person is in the room).
- Information from malicious extrinsic sources could feign a context that is not real. Because knowledge sharing needs a certain degree of openness it is easier for attackers to manipulate information even if it is obtained from sources within the own domain.

Anind Dey, who has co-written the most commonly cited definition of context ('any information that can be used to characterize the situation of an entity ... that is considered relevant for an interaction ... ') [1] has pointed out with others that: *... both sensed and interpreted context is often ambiguous. A challenge facing the development of realistic and deployable context-aware services, therefore, is the ability to handle ambiguous context.* [2]

Karen Henrickson et al. have determined as a characteristic of context information to be 'imperfect': *Information may be incorrect if it fails to reflect the true state of the world it models, inconsistent if it contains contradictory information, or incomplete if some aspects of the context are not known.* [3] They ascribe this to several reasons: Large delays between production and use of context information, crude sensors and disconnections between the context producer and the context consumer. Later on, therefore, they demand: *... context models will need to specify a range of characteristics of context information, including temporal characteristics (freshness and histories), accuracy, resolution (granularity), [and] confidence in correctness of context information ...* [4]

The management branch of our research team has analyzed management challenges in ubiquitous computing [5] also identify that data delivered by different context sources is consequently also very heterogeneous and usually varies in its update frequency (if any), its accuracy, and its format of representation. In [6] our research team has addressed the 'high risk of failures within the process of context provisioning' and proposed strategies to build missing context information out of others that might be available. The notion 'Quality of Context (QoC)' has at first been embossed and defined in 2003:

> Quality of Context (QoC) is any information that describes the quality of information that is used as context information. Thus, QoC refers to information and not the process nor the hardware component that possibly provide the information. [7]

The most important QoC-parameters have been listed as precision, probability of correctness, trust-worthiness, resolution and up-to-dateness [7].

3 Where QoC Matters

We have deduced requirements for modelling and handling QoC from analyzing QoC in the context provisioning process. So read in this section our understanding of this process.

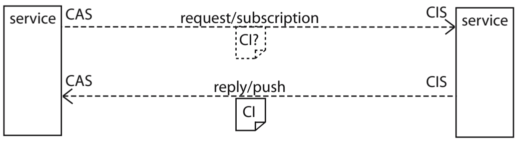


Fig. 1. Role model for the exchange of a context information object (CI) between a context-aware service (CAS) and a context information service (CIS)

3.1 Roles: CIS and CAS

Though we do not refer to any particular context model in order to keep these steps as general as possible, we define two service roles to describe our understanding of this process (see also fig. 1):

– A *a context information object (CI)* is a particular piece of typed context information.
– We furthermore define the role *context-aware service (CAS)* for any service requesting a context information object
– and the role *context-information-service (CIS)* for any service providing a context information object.

So the notion context information service stands for any type of context sources; amongst others this could be a sensor, a database, a profile base or a service that is able to aggregate a context information object out of other context information objects (e.g. a distance out of two positions). It is obvious that some services could adopt alternating both roles - context-aware service and context information service.

3.2 The Context Provisioning Process

In this paper we refer to any context provisioning process that suits the very abstract process description in figures 2 and 3 or is very similar to it. Because

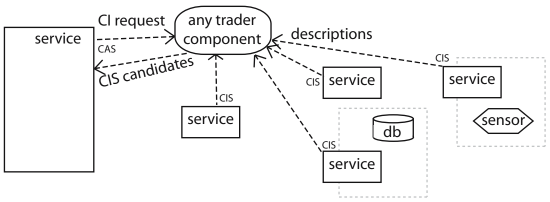


Fig. 2. A trading component arranges that the request of a context information object and context information service descriptions can be matched to discover an appropriate source

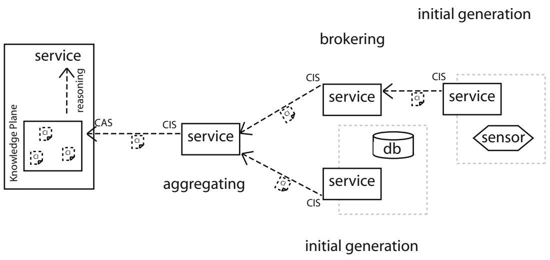


Fig. 3. An example value chain of context information objects. It is likely that a request for a context information object can not be fulfilled directly.

of the increasing mobility of users and devices we assume that a context-aware service has no hard-linked context sensors but has to search for context information services at runtime. Therefore these services must have published their service descriptions to any kind of trading component beforehand (see fig. 2). Here it is of no major importance whether this is implemented as a context information middleware, as a semantic query routing in peer-to-peer networks or whether this is an agent-based approach. We just assume that there *is* a trading component to manage the exchange of context information. Often the request for a context information object from a context-aware service can not be fulfilled directly. There might be the need to aggregate the context information object out of other context information objects and there might be other brokering context information services acting as intermediaries. Fig. 3 shows an example context value chain. After receiving the context information object the context-aware service can reason about all related context information he has already gathered. So it can form an own opinion about the actual context.

4 The Characterisitcs of QoC

4.1 Distinction Between Quality and Worth

From our discussions about QoC we know that there is a certain risk of talking at cross-purposes because of the term ‚quality‘. Quality can either be defined in an objective way. For example for the precision of temperature measurements of a sensor. Or the term can be used in a quite subjective way: A position information can be of high or low quality for a certain purpose. Therefore we want to concretize our notion of QoC:

Quality of Context (QoC) is any inherent information that describes context information and can be used to determine the worth of the information for a specific application. This includes information about the provisioning process the information has undergone (‚history‘, ‚age‘), but not estimations about future provisioning steps it might run through.

This distinguishes between the objective ‚quality‘ and the application-specific ‚worth‘ of a context information object. It also describes their relation: QoC is used to estimate the worth of a context information object for an application. This definition also implies the relation of QoC with the quality of service (QoS) of the provisioning process: Agreements between context-aware service and context information service about future provisioning steps of a context information object concern the QoS of the provisioning process. Information about the actual reached QoS in steps that are already done can become part of the QoC of the concerned context information object.

4.2 What Determines QoC

The QoC-values of a context information object can be affected at several steps within the provisioning process. At first there is the **initial generation** of a context information object. When an initially generated context information object is stamped with QoC-values there are several sources for this. We have identified: *the characteristics of the sensor, the situation of the specific measurement, the value expressed by the context information object itself, and the granularity of the representation format.*

We want to explain this list: That the characteristics of the sensor affect the quality of the result is obvious. Also important is the situation of the specific measurement. Think of how heat, coldness or humidity can affect physical sensors temporarily, or how heavy CPU-load can weaken real-time sensing abilities. Even the value of the information can have effect its own QoC. Just think of a position information that is expressed with the id of the radio cell the concerned entity is situated in. According to the size of the cell varies the precision of the position information. Above that it is possible that even the representation format effects QoC. For example the strength of a WLAN at a certain location could be expressed in rather imprecise categories of 'excellent, very good, good, average, . . . ', or it could be expressed more precisely in dBm (decibel milliWatt). Because this quality is inherent to the representation format it does not vary over the time.

The other source of effects to QoC-parameters is any kind of **transformation or brokering** a context information object undergoes during the provisioning. In detail: QoC-values might change with the conversion of the representation format of a context information object. The same does apply for the aggregation of several context information objects into a new one. (Just think of this example: I want to know the position of a PDA, but can not locate it. But I can sense its distance from a car that I can locate. So I can determine the position of my PDA roughly. The QoC of this position information depends on the QoC of the position information of the car, the QoC of the distance information and the accuracy of my geometric calculation.) Finally even the brokering path and the delivery time can be relevant QoC-values (relevant for a application to estimate whether a context information object is of use: It might already be too old, or might be passed on by context information services the CAS does not trust).

5 Challenges in Modelling QoC

5.1 Expressiveness of QoC-Parameters

When you begin to model QoC-parameters, even for very simple scenarios, you quickly learn: There is a wide range of possible representation formats to express a QoC-parameter. Labels like 'precision', 'granularity', 'probability of correctness' can only be coarse categories regarding to this variety. For example there is a endless range of possibilities for expressing ‚accuracy'. You can make statements about the average error, the minimal error, the maximum error, the probability distribution. You can express these values relatively or in absolute units – and so on. From this instantly follows the requirement for strong typed QoC-parameters. Complex information formats are necessary as well. Simple key-value-pairs, for example with symbolic values from 0 to 1 for ‚accuracy' are by far not expressive enough for open systems with a pervasive sharing of context information. Furthermore, which QoC-parameter could be applied to a context information object is heavily dependent on the context information class. To allow an automatic processing of QoC-parameters it must be predictable which QoC-classes a context information object can possibly have. This set is depending on the context information class and the provisioning process the object has run through (if it is a profiled, a sensed or a reasoned information and if it has undergone aggregation processes). Concerning their potential existence in a set there can be presuming and excluding dependencies among the QoC-classes. It is a design or modelling decision whether a context information class has a degree of freedom concerning a specific quality. Or whether the same is modelled as several context information classes each with this specific quality fixed inherently. For example: On the one hand a context information class 'bandwidth' with the time interval of the measurement as QoC-parameter, on the other hand a bundle of context information classes 'bandwith5seconds', 'bandwith5minutes' and so on with this time interval included inherently. Even if a context model applies a fixed set of QoC-parameters to a certain context information class there is no certainty that a context information service is able to supply them all with a context information object. As well as the high dynamic of context-provisioning implies the risk that context information is incorrect, incomplete or perhaps not available at all, the risk with QoC-values has the same status: Single QoC-parameter might be missing or incomplete.

Everything stated above in this section shows that modelling of QoC-values is very similar to modelling of value states of context information objects. Here a new problem arises: QoC threatens to start a recursion. Each QoC-parameter of a context information object could be understood as a context information object itself. So there is the question: what are the real differences between the representation of a QoC-parameter and the representation of a context information object itself? The automation of the context provisioning implies that also transformations of context information objects as well as conversions of representation formats can be handled automatically. This again means that adverse effects on QoC-values need to be predictable, otherwise the result could be a context information object that is no longer appropriate for the original request for a context information object. The question is, how far can statements about effects of conversions and transformations be integrated into context models. In any case this would make the mapping of ontologies far more difficult.

5.2 Modelling QoC Within Context Discovery

There is not only the need to express the values of QoC. As can be seen in the context provisioning process, before the context information object can be delivered there is the phase of context discovery that covers the matching of **the context information service description with the request for a context information object**. To specify the request for a context information object a possibly wide variety of descriptive constraints like thresholds and intervals are possible. Possible among constraints are also dependencies, even functional dependencies and priorities. To increase the chance of finding appropriate context information some constraints should be optional. If a context-aware service requests a context information object immediately (‚pull‘) it is able to state its conditions in an absolute manner. But if a context-aware service sends a subscription for context information objects that are to be delivered in case of certain events (‚push‘) it might express further dynamic QoC-dependencies in advance. Can such dependencies still be expressed in a descriptive way? Or does this mean an overloading of a context- and QoC-model? The context information service description should be used to decide whether a context information service can provide appropriate context information objects for a certain request. But this is only correct for QoC-parameters that depend only on the representation format and the sensor characteristics. QoC-parameters depending on the measurement situation might change too fast to make predictions possible. And if some QoC-parameters are also depending on the respective context information object itself (e.g. the width of a radio cell for cell-based positioning) there can hardly a prediction at all. This makes it difficult to decide beforehand whether a context information service can deliver appropriate context information objects. The description of a context information service falls into three parts. A static part that can easily be published, a dynamic part, that could at least be retrieved shortly before the measurement, and a purely prediction part.

5.3 Using Context with QoC

Besides the QoC-information described above that is inherent to the information and can be stated without respect to the use, there is the application-view on a context information object. An application uses own estimations (‚worth‘) that go beyond inherent QoC to select among available context information services and context information objects as well as to manage obtained context information objects or to assess them for reasoning.

Reputation and Trusted Third Parties. It is not only the numb comparison of the request with the QoC of the actual retrieved context information object that decides about how ,good‘ it suits an application. The application additionally needs to decide, how much faith it puts in the correctness. This depends on the reputation of certain context information object and the reputation of the context information service it comes from. The context information object itself can be approved in two ways: The absence of contradictions and the presence of consistence with other context information (from other sources) about the same context. A context information service can build up a good reputation in the view of an application, if the application has made good experiences with it in the past, or if it learns from other (trusted) context-aware services that have made good experiences. As it is reasonably to share the information about your trust in a context information with others who put faith in you, it seems sensible to find ways to express this trust in an exchangeable way.

Storing Context Information Objects. Context-aware services are likely to store obtained context information objects for reuse and to gain a better image of the current situation. It seems reasonable to annotate the stored information regarding to their temporal validity and their trustworthiness from the viewpoint of the service. The annotation of the temporal validity accelerates the decision whether a context information object can still be subject of a valid reasoning and it triggers if necessary the anew fetching of a context information object. The question is in how far it is reasonable to store these subjective information in the same structures as the QoC-values that are application-independent. As these information are application-dependent it seems appropriate to indicate in each case which respective application they are from.

6 The Context Metamodel (CMM) and Outlook

In our own 'Metamodel Approach‘ (for a more detailed insight please see [8]) to modelling context we have included a base construct to represent quality aspects of context information (fig. 4). It is not of the same class as the context information value (this is represented by the metaclass 'DatatypeClass'), but uses the same data constructs and transformation rules. With this parallelism between QoC- and context-information-modelling we meet most requirements we have made concerning the expressiveness of QoC (see 5.1). The Context Meta-Model (CMM) is realized as information model inside the Java-based ,CoCo

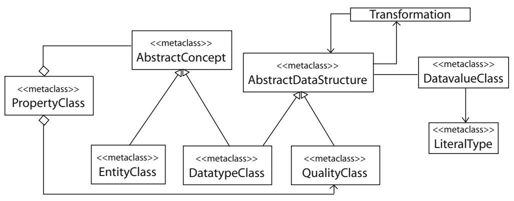


Fig. 4. Simplified overview of the Context Meta-Model (CMM)

infrastructure‘, a middleware for providing composed context information. Nevertheless we have not realized the use of QoC within context discovery and have to investigate how QoC can be used in requests for context information objects and context information serive descriptions. The modelling and use of reputation information and application-specific meta-information is also an open issue.

7 Related Work: QoC in Context Models

The **Nexus Platform** is an open pervasive computing system that aims to provide a ’federated, potentially global context model’ [9]. Its research team proposes a context model that consists of typed objects composed of different attributes. The objects are stored at several data providers. They attach meta data to objects (like lifetime, date of first data gathering or data about the initiator of this object), attributes (like valid time, gathering time, accuracy or measuring system) and to providers (like trustfulness and access costs). Nexus works with simple key-value-pairs that are not annotated with units. Their notion of ’accuracy’ is as coarse that it works with a $[0 - 1]$ range for both temperature expressed in (apparently) Celsius and position expressed in terms of longitude and latitude. This is not suitable for the requirements we have made for a more sophisticated reasoning and a automated handling of a context information object. The ontology-based context model for the **Socam Architecture** [10] allows associations (OWL properties) between properties of entities and quality constraints indicating QoC. Four types of most common quality parameters are defined: accuracy, resolution, certainty and freshness. A metric consists of the triple (unit, type, value). At least these key-value-pairs are annotated with a unit, but still do not allow complex data formats. Every QoC-type theoretically can be attached to every type of property - there are no further restrictions or dependencies possible in this model. The **ASC model** [11] allows complex data types because and defines every quality of a context information as context information itself. Unfortunately the ASC model does not allow the modelling of dependencies between context information classes and quality classes. The applicability of the model is also limited because of a disarranged use of concepts and instances. The **Object Role Modeling (ORM)** of context has been enhanced [12] with constructs to describe probability (as a number) and to state whether the source of a fact is static, sensed, profiled or derived whereas these categories presume different quality issues. This approach lacks expressiveness of QoC-values.

8 Conclusion

We analyzed the context provisioning process and emphasized the need for QoC. By distinguishing between context-information-inherent QoC and application-specific meta-information hopefully we rendered more precisely the notion of QoC. With a summary of requirements for modelling and handling QoC we want to share our ideas with other research teams. In any case we demand a modelling of QoC that is far more expressive than recent approaches.

References

1. Dey, A.K., Salber, D., Abowd, G.D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction (HCI) Journal **16 (2-4)** (2001) 97–166
2. Dey, A.K., Mankoff, J., Abowd, G.D., Carter, S.: Distributed mediation of ambiguous context in aware environments. In: Proceedings of the 15th Annual Symposium on User Interface Software and Technology (UIST), Paris, France (2002) 121–130
3. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In Mattern, F., Naghsineh, M., eds.: First International Conference on Pervasive Computing (Pervasive). Number 2414 in Lecture Notes in Computer Science, Zurich, Switzerland, Springer (2002) 167–180
4. Indulska, J., Robinson, R., Rakotonirainy, A., Henricksen, K.: Experiences in using cc/pp in context-aware systems. In: 4th International Conference on Mobile Data Management (MDM). Number 2515 in Lecture Notes in Computer Science, Melbourne, Australia, Springer (2003) 247–261
5. Hegering, H.G., Küpper, A., Linnhoff-Popien, C., Reiser, H.: Management challenges of context-aware services in ubiquitous computing environments. In Brunner, M., Keller, A., eds.: Self-Management Systems: 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM). Number 2867 in Lecture Notes in Computer Science, Heidelberg, Germany, Springer (2003) 246–259
6. Hochstatter, I., Krause, M.: Strategies for on–the–fly composition of context information services. In: Proceedings of the 11th International Workshop of the HP OpenView University Association (HPOVUA). Volume 2004., Paris, France (2004)
7. Buchholz, T., Küpper, A., Schiffers, M.: Quality of context information: What it is and why we need it. In: Proceedings of the 10th International Workshop of the HP OpenView University Association (HPOVUA). Volume 2003., Geneva, Switzerland, Hewlett-Packard OpenView University Association (2003)
8. Fuchs, F., Hochstatter, I., Krause, M.: A metamodel approach to context information. In: Proceedings of the 3rd International Conference on Pervasive Computing and Communications (PerCom) Workshop, IEEE Computer Society (2005)
9. Hönle, N., Nicklas, U.P.K.D., Schwarz, T., Grossmann, M.: Benefits of integrating meta data into a context model. In: Third IEEE Conference on Pervasive Computing and Communications Workshops – Workshop on Context Modeling and Reasoning (CoMoRea), Kauai Island, Hawaii, USA (2005)
10. Gu, T., Wang, X.H., Pung, H.K., Zhang, D.Q.: An ontology-based context model in intelligent environments. In: Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, San Diego, USA (2004)
11. Strang, T., Linnhoff-Popien, C., Frank, K.: Cool: A context ontology language to enable contextual interoperability. In Stefani, J.B., Demeure, I., Hagimont, D., eds.: 4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS). Number 2893 in Lecture Notes in Computer Science, International Federation for Information Processing (IFIP), Springer (2003) 236–247
12. Henricksen, K., Indulska, J.: Modelling and using imperfect context information. In: Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, Orlando, Florida (2004)

Secure and Scalable Routing Protocol for Mobile Ad-Hoc Networks*

Namhi Kang[1], Ilkyun Park[2], and Younghan Kim[2]

[1] Ubiquitous Network Research Center in Dasan Networks Inc., Yatap-Dong, Bundang-Ku, 463-070 Seongnam, South Korea
nalnal@dcn.ssu.ac.kr

[2] University of Soongsil, School of Electronic Engineering, Sangdo 5-Dong 1-1, Dongjak-Ku, 156-743 Seoul, South Korea
yhkim@dcn.ssu.ac.kr

Abstract. This paper is intended to find an efficient way to support secure and scalable routing mechanism well suited for Mobile Ad hoc Networks (MANET). Support of secure routing protocol is prerequisite for launching a secure communication in the presence of adversaries. To do this, we first propose a zone networking architecture based on zone masters that allow ordinary nodes to discover a feasible route to the destination in a scalable manner. On the basis of the architecture, secure routing protocol is given. Moreover, we also propose to use short-lived key(s) to overcome the performance degradation caused by applying expensive digital signature to our scheme.

1 Introduction

In MANET, the communication only relies on mutual and cooperative routing functionalities of ordinary nodes without any specific relaying devices such as a router in a wired network. Hence, it is hard to directly employ conventional routing protocols used in wired networks into MANET. For this reason, lots of routing protocols have been proposed in the literature [1]. However, there still exists severe performance degradation when the network size increases [2]. Moreover, such routing protocols are vulnerable to many kinds of attacks [3]. It is fairly easy to inject fake routing messages or modify legitimate ones so that network operation would be heavily disturbed (e.g., by creating loops or disconnecting the network). Therefore, it is necessary to find a secure and scalable routing protocol for a large scale and densely populated MANET.

In this paper, we first propose a *"zone based network architecture"* to support a large scale MANET. The proposed architecture borrows the basic concept of cluster based routing protocols but we remove the highly expensive processing to elect a cluster head. Instead, each cluster (referred to as a zone in our approach)

* This research is supported by the ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

has a *zone master* (ZM). A ZM gathers routing information and distributes it to nodes within his zone so that the nodes can discover and maintain a route at a low cost. As a result, we can dramatically reduce lots of overburden of nodes and flooding of control packets necessary to maintain a cluster and its head.

Secondly, we propose "*secure and scalable routing protocol (SSRP)*" which is intended to protect against falsely modified routing information generated by malicious node(s). If there is no means for supporting route discovery in a secure manner, the efforts to find a proper way to protect data transmission itself may be a good-for-nothing. For example, an attacker may manipulate the routing process to let him become a relay node for malicious handling of in-transit data; a selfish node may manipulate the route discovery process to exclude him from a route for saving his energy; and an adversarial node may try to impersonate the other node who can be an initiator or a replier of the route discovery process.

Finally, we propose an efficient way to support data origin authentication of routing information using a *"short-lived but secure key"*. In SSRP, ZM digitally signs on routing information to broadcast it periodically to his member nodes. A drawback of regular digital signature schemes such as RSA or DSA is the performance problem. Such a problem is generally caused by using a large key size for protecting data in the context of long-term security. If one requires a protection against attacks only for a short period such as a week or a day, then using a key which is long enough for one or two decades is a waste of resources.

This paper is organized as follows. In section 2, we discuss MANET routing protocols in terms of the scalability and security. The proposed architecture and SSRP are given in section 3. In section 4, we propose a way to select a short-lived key in a secure fashion. In section 5, we analize the proposed approaches. Finally, we conclude this paper in section 6.

2 Related Work

2.1 Scalable Routing Protocol

Scalable routing protocols in MANET are categorized into two [2]: one is cluster based hierarchical routing protocols such as zone routing protocol (ZRP) [4] and the other is geographical location based routing protocols such as location-aided routing (LAR) [5]. In the former, a MANET is logically partitioned into a set of clusters. Thereafter routing procedures are performed in each cluster separately to manage a large scale of MANET. On the other hand, the latter utilizes geographical position information for routing under the assumption that all nodes are equipped with global positioning system (GPS) module. Therefore, the former seems to be more suitable for MANET since it is more or less impractical for all nodes to be equipped with GPS module.

Cluster based protocols, however, still introduce high overhead because all of nodes must exchange information to elect a cluster head in addition to the routing procedure itself. Furthermore a cluster head should be reelected whenever the network topology of a cluster is changed. In some scenarios, the topology is frequently changed by the mobility of nodes in MANET. Further, a cluster head node consumes much energy than other nodes because the head has to do concentrative computing. If a cluster head is disappeared abnormally, all of nodes in a cluster can not communicate with each other during some periods of time required for reelection of a head. To overcome these problems, ZRP was proposed. ZRP does not require any procedures for electing a cluster head. Instead, each node becomes a head and forms a localized routing region defined as a zone. ZRP does not specify a certain protocol but rather it applies a proactive routing protocol for nodes within a zone and a reactive routing protocol for nodes belonging to different zones in a hybrid manner. Thus, ZRP takes advantages of both proactive and reactive routing protocols.

2.2 Secure Routing Protocol

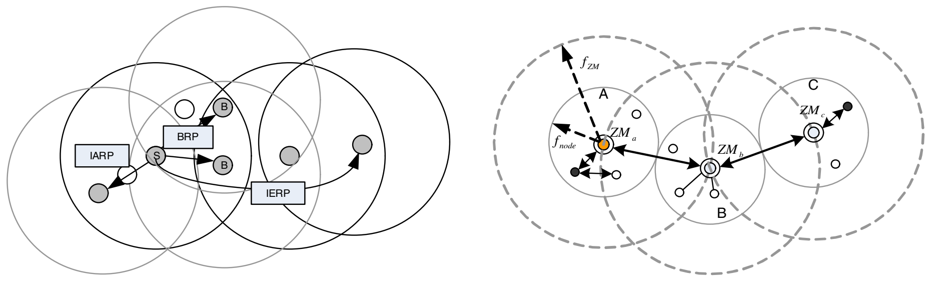
It is difficult to support secure routing in MANET due to the lack of a priori trust model in which legitimate routers perform correct operations [3]. To overcome such a challenge, several schemes have been proposed recently.

SEAD is based on the DSDV routing protocol [6]. Hence, SEAD can be used in a limited scale of MANET. Unlike SEAD, ARAN [7] and ARIADNE [8] are a reactive secure routing protocol. In ARAN, however, nodes (including the source and the destination) along the route must verify the signature that the previous node of the path generates and also sign on RDP (Route Discovery Packet) or REP (Route Reply Packet) packet after appending its certificate to the packet. That is, $n + 1$ times signing/verification operation are required for a connection linked by n hops. In some cases where nodes have limited processing power, ARAN is impractical to be used because of its highly expensive overhead. Hu et. al. proposed a secure routing protocol based on DSR called ARIADNE. They employ TESLA, which is a source authentication scheme for streaming over multicast [9], to provide authentication of a routing message from a point to point. ARIADNE has better performance than ARAN since it utilizes a symmetric cryptography. A drawback of the scheme is that all nodes in networks should be time synchronized. Even though the synchronization is done in a loosely manner, such a requirement may be impractical in a large scale network. They also propose to use a shared key between each pair of nodes so as to avoid the need for synchronization. But such a way requires very high overhead for key setup.

3 Proposed Architecture and SSRP

3.1 Zone Based Network Architecture

Fig. 1 presents the proposed architecture (in (b)) in comparison with ZRP (in (a)), where we consider a relatively large and densely populated MANET. In ZRP, each node forms its own zone resulting in a large number of overlapping of nodes. This leads to increased control overheads even though a larger number of neighbors per node are required for robust connectivity. The proposed scheme reduces such overlapping regions by utilizing ZM.



(a) Zone routing protocol (b) Proposed network architecture

Fig. 1. Zone based MANET

MANET is divided into several routing zones in which a ZM is assigned for each zone. Nodes within a predefined zone radius are referred to as member nodes. Unlike member nodes, ZM has high computing power and robust electrical power as a super node (neither a source nor a sink). ZM maintains consistent and up-to-date routing information so that member nodes do not need to follow the changing of network topology. That is, highly expensive computing required to generate and maintain a routing table is performed only by ZM.

In order to achieve a robust connectivity between nodes, ZM may be allowed to adjust his transmitting power differently (namely, different radio level). As shown in (b) of Fig. 1, ZM uses two different powers, f_{node} and f_{ZM}. The signal power for transmitting data between ZMs should be higher than power for sending data from ZM to member nodes (i.e. $f_{node} > f_{ZM}$). The wireless backbone formed by linked ZMs provides robustness against a scenario where no boarder nodes, that play a role in relaying a packet between zones in ZRP, exist in an overlapping region. Hence, packets can be transmitted via the wireless backbone without aid of boarder nodes. If the wireless backbone is not used, traffic may concentrate on just a few nodes resulting in a long end-to-end delay due to congestion at the nodes and a high energy consumption of the nodes. Therefore, it may lead to network isolation and degradation of lifetime of entire ad hoc networks.

As a result, we can reduce lots of overhead necessary to elect a cluster head whenever the topology is changed; we can remove the single point of failure problem caused by energy exhaustion of a cluster head node; and we can save resources in energy and data processing of ordinary nodes in MANET by maximizing the role of ZM.

3.2 Intra Zone SSRP

ZM proactively integrates link information of his zone into an IntLS (Integrated Link State) packet. Thereafter, the ZM signs the IntLS packet and then broadcasts it to member nodes periodically. We assume that every node has ascertained the correctness of a public key corresponding to a private key used for signing operations in advance by use of key management protocol and public key certification mechanisms such as PKI (Public Key Infrastructure). Member nodes in a zone are thus capable to communicate with each other directly by using information containing in the IntLS packet. The information includes the list of zone member nodes and TTL value which is used to restrict a broadcasting region of IntLS packets within a zone. The main difference from other routing protocols, only ZM updates link information of the zone in our scheme.

The problem of the proposed intra zone SSRP is the computational cost necessary to digitally sign on IntLS packet at the ZM as well as to verify the signed packet at all nodes. However, if data origin authentication is only the context, we can reduce the length of a key (see section 4 in detail). This is mainly due to the fact that the life time of authenticated link information is short. Namely, a member node verifies an IntLS packet only to assure that the link information was originated by the claimed ZM and thus the packet was not modified in transit. We note here that non-repudiation of origin is not necessary for our case.

3.3 Inter Zone SSRP

A node checks his routing table when he has a packet to send. If there is no information in the table, the destination may belong to other zones or be newly moved into the zone just before the routing table update. The source sends a route discovery (RD) packet to his ZM to discover a route to the destination. A ZM that takes the RD packet checks whether the destination is one of his member nodes. If the ZM finds the node, he digitally signs on a newly updated IntLS packet including information of the new member node then broadcasts it. Otherwise, the ZM relays the RD packet to his neighbor ZMs.

In order to authenticate this RD packet, a ZM appends authentication tag to the packet in two different ways. The first way, on one hand, is that the ZM utilizes digital signature to generate the tag. On the other hand, the ZM can use a MAC primitive such as HMAC with a shared key between ZMs. To use the latter way, a single shared key should be distributed to all ZMs through a secure channel. An administrator of the network may configure all

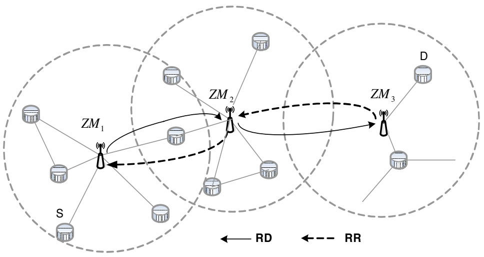


Fig. 2. Inter-zone SSRP

ZMs in advance. Alternatively, any key management scheme can be employed to distribute a single key to all of ZMs. The trade-off is that the latter is much efficient than the former in terms of the computational cost but requires an additional mechanism to manage the security association between ZMs. Fig. 2. shows the route discovery procedures as an example.

In the figure, it is assumed that the source, S, which belongs to ZM_1's zone has a packet to send to the destination, D, in the ZM_3's zone. S searches the destination in his routing table which is based on the information of IntLS packet. Because the destination does not exist in the table, S generates a route discovery packet, RD, to send it to ZM_1. The ZM_1 generates authentication information then appends it to RD. Thereafter, intermediate ZMs (here ZM_2) verify RD. If RD is authentic, then they relay the packet to neighbor ZMs. Otherwise, RD is discarded. Finally, ZM_3 generates a route reply packet, RR to send it back to the S. RR packet also conveys authentication information.

4 Short-Lived Key Selection

To reduce the overhead of applying digital signature, a ZM generates a signature using a short key of length which is available only for a predefined short duration. The challenge is thus how to derive an appropriate length of key without compromising the security. Our approach is based on the lower bound of the Lenstra and Verheul's suggestion (LV's recommendation for short) [10].

4.1 Overview of LV's Recommendation

In [10], the authors recommended appropriate key sizes for both symmetric and asymmetric cryptosystems based on a set of explicitly formulated parameter settings combined with historic data points of attacking on the cryptosystems. They defined the $IMY(y)$ as a MIPS-Y (Million Instructions Per Second-Years) that is considered to be infeasible until year y to derive a key size necessary to support computationally equivalent to the same strength offered by the DES in 1982. The computational efforts of 5×10^5 Mips-Y is considered as a secure cost to resist software attack on commercial DES in 1982. $IMY(y)$ is formulated by

$$IMY(y) = 5 * 10^5 * 2^{12(y-s)/m} * 2^{(y-s)/b} \text{ MIPS-Y}, \tag{1}$$

where s, m and b are variables used to consider environmental factors that are able to affect the security in the selection of key length.

The definitions and its default settings (value in the round bracket) of such variables, namely the explicit parameter settings of their guidelines, are as follows; $s(= 1982)$ is defined as the year until which a user was willing to trust the DES, $m(= 18)$ is defined as the number of months it takes on average for and expected two-fold processor speed-up and memory size increase, and $b(= 10)$ is defined as the number of years it takes on average for an expected two-fold increase of budget. The default setting for m and b are derived from the Moore's law and the US Gross National Product respectively.

On the basis of equation (1), to select the length of the conventional asymmetric key system, they applied the asymptotic run time L[n] of a NFS (Number Field Sieve), which is the best known technique for factoring large number, combined with a historical fact that a 512 bits modulus was broken in 1999 at the cost of around 10^4 MIPS year, such that

$$\frac{L[2^k]}{IMY(y) * 2^{12(y-1999)/18}} \geq \frac{L[2^{512}]}{10^4}, \tag{2}$$

where the $o(1)$ factor is omitted from the run time $L[n]$.[1] In equation (2), a factor $2^{12(y-1999)/18}$ indicates that the cryptanalytic is expected to become twice as effective every 18 months during year 1999 to year y.

4.2 Short-Lived Key Selection

In order to derive a short-lived key pair, we define $IAO(x, y)$ as the *'Infeasible Amount of Operations'* within a certain period, x, of the year y. $IAO(x, y)$ is derived from the $IMY(y)$ formulated in equation (1). $IAO(1\ year,\ this\ year)$ is therefore derived from $IMY(this\ year)$. Here, it is emphasized that the term *'MIPS'* is used as a measuring tool for the relative comparison of computational workload thus the trend of value, for example ratio of increasing/decreasing, is important rather than its absolute value. Like [10], it is also supposed that a single operation requires a single clock cycle. That is, a 2.4 GHz Pentium 4 computer (which is used for our experiments) is regarded as a 2400 MIPS machine. A short-lived key length is calculated based on the following intuitive theorem 1.

Theorem 1. *Let $S = \{x_1, \cdots, x_j : x_1 + \cdots + x_j = X\}$ be a set of sub periods of X, and k be the length of a short-lived key. If k meets equation (4), then the length of k is secure enough within x_i, for all $x_i \in S$.*

$$L[2^k] \geq 55.6 * 2^{12(y-1999)/18} * IAO(x_i, y). \tag{3}$$

Proof. Suppose that a computer operates OPs during period X as the maximum amount of operations. If that is the case, the maximum number of operations of the computer in period x_i is $OPs * (x_i/X)$ under the assumption that an operation is done in a clock cycle, where x_i is one of the sub periods of X. That is, all elements of S have their own $IAO(x_i, y)$ and the total amount of these must be equivalent to $IAO(X, y)$ since $\sum_{i=1}^{j} IAO(x_i, y) = \sum_{i=1}^{j} IAO(X, y) * \frac{x_i}{X} = IAO(X, y)$. Therefore, $IAO(x_i, y)$ is an infeasible amount of operations within x_i in the year y.

Now we can derive equation (3) by applying $IAO(X, y)$ to equation (2) since $IMY(y) = IAO(X, y)/(3.15 * 10^{13})$, where $X = 1\ year$ and 1 MIPS year is equivalent to $3.15 * 10^{13}$ operations. We approximate here that $\frac{L[2^{512}]}{10^4}$ is equivalent to $1.75 * 10^{15}$. Finally we get equation (3) since $IAO(X, y) * \frac{x_i}{X} = IAO(x_i, y)$.

[1] $L[n]$ is an abbreviated form of $L[n, u, v] = e^{(v+o(1))ln(n)^u ln(ln(n))^{1-u}}$, where u is 1/3 and v is 1.9229 in NFS.

Consider 2004 as the year y of $IMY(y)$, for example, then it is expected that $IMY(2004)$(=$5.98 * 10^{10}$ MIPS) is secure until the end of 2004 according to the equation (1). This is equivalent to $1.88 * 10^{24}$ operations (OPs) thus such a number of operations are infeasible until the end of 2004, namely $IAO(1\,year, 2004)$ is $1.88 * 10^{24}$. Thereby we can calculate a proper key length for any sub period as described in Table 1. If a day is determined as a short period, the source can use an 880-bit key to sign. This is because $IAO(1\ day,\ 2004)$ is $5.16 * 10^{21}$ which is equivalent to $1.64 * 10^{8}$ MIPS year. 880-bit can be derived by using equation (3). The signature computed with an 880-bit key offers the same security level for a day as does a 1108-bit key for this year and a 2054-bit key for the next two decades.

Table 1. Short-lived key length selection and its computational cost

Period x_i	$IAO(x_i,\ 2004)$	Key length (bits)	RSA (msec) sign/verify	DSA (msec) sign/verify
1 decades	$3.81 * 10^{26}$	1562	29.31 / 0.65	6.13 / 9.31
1 year	$1.88 * 10^{24}$	1108	7.92 / 0.31	3.64 / 4.15
1 Month	$1.57 * 10^{23}$	1008	5.92 / 0.29	2.87 / 3.14
1 Week	$3.67 * 10^{22}$	952	5.37 / 0.27	2.71 / 2.82
1 Day	$5.15 * 10^{21}$	880	4.55 / 0.23	2.34 / 2.71

To experiment such computation time, we use Crypto++ library [11] on a Linux based 2.4GHz Pentium 4 computer, where the input size is 1024 bytes. In RSA ($e = 2^{16} + 1$, $|d| \approx |n|$), the key size means the modulus length (n), while the key size of DSA ($|q| = 160bits$) indicates the field size (p).

5 Protocol Analysis

5.1 Security Analysis

The zone radius (the number of hops) highly affects the performance of MANET in our approach. Larger zone radius gives more benefits to the case where the mobility of nodes is very low (e.g. in conference room), while increased mobility of nodes requires a small zone radius to take advantage of the reactive routing protocol. From the secure point of view, a single hop is the best solution. In the case of a zone covering nodes in multiple hops, it is difficult to protect against the selfishness problem and insider coalition to deceive other nodes by inserting bogus routing information as described in Fig. 3. A selfish node may discard RD packets of a claimed node to exclude him from a route for saving his battery. In addition, two malicious nodes may insert bogus routing information to cheat other nodes (C and D in (b) of Fig3). It is difficult for ZM to recognize this cheating. To protect against such attacks, we restrict a zone radius to one hop so that all member nodes are capable for connecting with his ZM directly. The transmission range specified in IEEE 802.11b is 250 meters. The maximum region

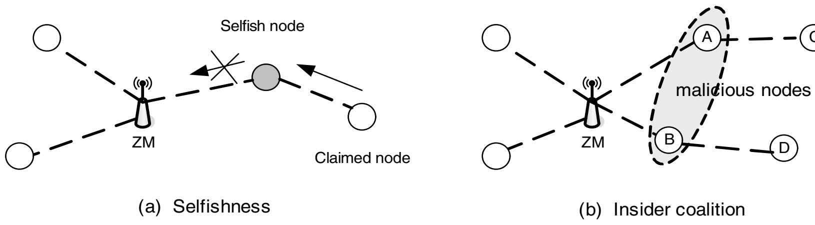


Fig. 3. Attack examples in a zone

that a hop can cover is 500 meter in diameter or around 196 square kilometers in space; thus a few tens of ZMs are enough to cover a town in scale for example.

Further, the proposed scheme allows a user to cope with the wormhole attach which is regarded as a crucial hindrance in designing secure routing protocol in ad-hoc networks. A malicious node who tries to mount a wormhole attack copies packets at one place in the network thereafter he/she movies them into the other place by means of a tunneling through the networks with his cooperating node. Such an attack can easily disrupt a correct route to the destination. In our scheme, every control packets that are related to the routing procedure are from a ZM. In addition, ZMs that are located in the network are linked together to form a backbone, that is both data packets and control packets are transmitted via ZMs. Hence, if all ZMs are of trust and well authenticated with each other then the wormhole attack is highly difficult to be mounted.

5.2 Performance Analysis

We evaluate the performance of the proposed scheme in comparison to ZRP using the NSv2 simulator. In simulation, the moving speed of each node is selected randomly from 0 to 3 m/sec; twenty five randomly selected sources transmit a packet of 128 bytes in size at every 0.5 second; and four ZMs are assigned uniformly at the space of 1500x1100m in size. In Fig. 4, (a) presents that the

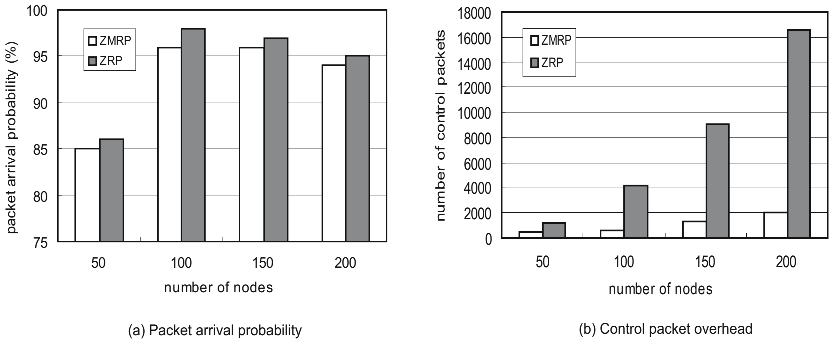


Fig. 4. Simulation results

probability of packet arrival at the destination correctly of both protocols is similar to each other. However, as shown in (b), the proposed scheme generates less control packets than ZRP because only ZM node broadcasts IntLS packets to member nodes, while each node of ZRP should broadcast its link information to other nodes.

6 Conclusion

We proposed a secure and scalable routing protocol based on the zone networking architecture. In our approach, an attacker cannot impersonate a ZM so that he cannot modify the correct routing information and also he cannot replay or insert stale routing information in transit. In addition, the result shows that the proposed scheme is efficient in terms of the computational cost as well as the transmission overhead.

References

1. E. M. Royer, S. Barbara and C.K. Toh. A Review of Current Routing Protocols for Ad hoc Mobile Wireless Networks. IEEE Personal Communications, pp. 46-55, 1999.
2. X. Hong, K. Xu and M. Gerla. Scalable Routing Protocols for Mobile Ad Hoc Networks. IEEE Network Magazine, Vol.16, pp. 11-21, 2002.
3. P. Michiardi and R. Molva. Ad hoc network security. ST Journal of System Research, Vol. 4, N1, Mar. 2003.
4. Z. J. Haas and M. R. Pearlman. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. IETF Internet Draft, 1998.
5. Y. B. Ko and N. H. Vaidya. Location-aided Routing (LAR) in Mobile Ad Hoc Networks. ACM/IEEE Int'l. Conf. Mobile Comp. Net., pp. 66-75, 1998.
6. Y-C Hu, D. B. Johnson and A. Perrig, SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks, In Proc. of IEEE WMCSA, Jun. 2002.
7. B. Dahill, B. N. Levine, E. Royer and C. Shields. ARAN: A secure Routing Protocol for Ad Hoc Networks, UMass Tech Report 02-32, 2002.
8. Y-C Hu, A. Perrig and D. B. Johnson. Ariadne : A secure On-Demand Routing Protocol for Ad Hoc Networks. In proc. of MOBICOM, 2002.
9. A. Perrig, R. Canetti, J. D. Tygar and D. Song, Efficient Authentication and Signing of Multicast Streams over Lossy Channels, IEEE Security and Privacy Symposium, May. 2000.
10. Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. Journal of Cryptology 14(4):255-293, 2001.
11. Crypto++ class library, http://www.eskimo.com/ weidai/cryptlib.html.

RAPOSI: Rapidly Installable Positioning System for Indoor Environments

Florian Schreiner and Holger Ziemek

Fraunhofer FOKUS, Next Generation Network Integration,
10589 Berlin, Germany
{Schreiner, Ziemek}@fokus.fraunhofer.de
http://www.fokus.fraunhofer.de

Abstract. RAPOSI is a radio-signal-strength based positioning system for indoor environments. Independent self-localization as well as centralized tracking of light-weight mobile devices is enabled. By omitting typically required a-priori scene analysis, set-up time is minimized. The positioning strategy is propagation-model based, retrieving and filtering radio-signal strength measurements from as many neighbouring reference points as possible and mapping them to distances. In order to increase reference-point densities, RAPOSI integrates several wireless technologies, namely 802.11 WLAN, Bluetooth and RFID. The results emphasize the benefits of hybridization, describe technology-specific neighbourhood-detection mechanisms, evaluate positioning algorithms, highlight advantages of adaptive position smoothing filters and finally evaluate overall positioning accuracy. Essentially important for the performance of the positioning system is the specific choice of the integrated RFID technology. The benefits of long-range semi-active UHF RFID tags that propagate their position and whose radio-signal-strength can be measured continuously, heavily account for the performance of RAPOSI.

1 Introduction

Accurate localization of mobile devices has been an active area of research for several years. Although the global positioning system (GPS) already enables a broad spectrum of location based services outdoors, still there is no commonly accepted technology that enables high precision localization indoors. Nevertheless location sensing technologies represent a core component of ambient, ubiquitous, pervasive computing architectures, highly interesting for many business cases as well as emergency/security scenarios.

High cost, complexity, rarely met infrastructural prerequisites, administrative difficulties, privacy issues and/or weak accuracies are usually the main reasons, why indoor positioning systems are yet not being commonly deployed in a broad scale.

In depth surveys have shown that, already, there exists a high demand for sub-100m accuracy positioning systems. They have shown that in order to kick-start mass-market adoption, positioning systems have to be extremely cost effective. Many already developed solutions (like video-, ultrasound- or physical contact-based approaches), however, are neither affordable for location-based service providers nor for their possible clients or customers.

Cell-based approaches like cell-of-origin or enhanced observed time-difference of arrival based approaches, although cost effective and scalable, lack indoor positioning accuracy. The same applies to global positioning system based methodologies since even with assisted or differentiated GPS, highly accurate indoor position estimations can not be delivered.

Therefore, in order to deliver a *cheap and broadly applicable* LBS-enabling solution we concentrate on commonly integrated wireless communication technologies (i.e. Bluetooth, WLAN) and on inexpensive off-the-shelf RFID tags.

Concerning reliability, robustness and accuracy, it shows that a *combination of different wireless interfaces*, promises to yield best results. Especially the combination of a far-range network enabled wireless transmission technology (WLAN, Bluetooth Class III) for continuous tracking, and a short-range beaconing methodology with cheap tags / beacons (Bluetooth Class I, RFID) delivers promising results. This is not only true in terms of cheapness of components, but also in terms of accuracy, since far-range indoor tracking methodologies deliver fair granularity of room-size resolution, whereas short-range beacons deliver fine granularities (i.e. high localization resolutions). Moreover, being connected to a network, beacons need only store and transmit a short unique amount of data, which is mapped to arbitrarily big amounts of data (stored on entities on the network, e.g. on the Internet).

The positioning system RAPOSI, represents the successor of two earlier designed systems. We first concentrated on lower layered mechanisms (RSS retrieval, filtering and propagation model acquisition) in order to develop a system, open and modular for the integration of forthcoming radio based technologies like WiMAX, Hiperlan or ZigBee. The outcome was ORPS [1], the Omni Radio-based Positioning System for indoor environments. Successively, we investigated adequate positioning mechanisms (Multilateration-, Trigonometric- and Mass-Spring-Model-based algorithms). The results determined our second prototype, GRIPS [2], a Generic Radio-based Indoor Positioning System. RAPOSI integrates lower layered modules from ORPS [1], the Mass-Spring Model, investigated in GRIPS [2] and by applying appropriate Kalman-filters, the first run-time evaluation of RAPOSI's performance is discussed in this paper. Furthermore, the benefits of an infrastructural independent and rapidly installable positioning system are explained, by illustrating RAPOSI's simple, rapid and efficient set-up schema.

The next section describes RAPOSI's relation to current indoor positioning strategies, highlighting differences and similarities. In *chapter three* general design of RAPOSI is illustrated, explaining the modules of each architecturally independent layer. In *chapter four* the layered architecture of the positioning system is being illustrated. The sensor layer, where different radio interfaces detect neighboring devices/tags and measure RSS of incoming signals is discussed in *chapter 4.1*. Subsequently, at the measurement layer in *chapter 4.2*, measurement and conversion of RSS into meaningful units that can be related to distances, with the help of appropriate radio propagation models, is elucidated. Thereafter in *chapter 4.3*, these measurements are collected at the fusion layer, where they are weighted, filtered and refined, in order to determine the best fitting position. Finally, in *chapter five*, the paper is concluded.

2 Related Research and Innovations

In contrast to existing indoor localization systems, RAPOSI *does neither depend on expensive positioning hardware* (like Video, UWB or Ultrasound like in [3],[4]*) nor on time-consuming, non-reusable environmental profiling* (extensive radio signal strength measurements at many different locations in the building like in [5],[6]). Therefore, RAPOSI is both: quickly deployable, as well as cheaply utilizable; hence RAPOSI is applicable in many indoor scenarios, enabling location based services in various different situations (emergency-scenarios as well as standard office/museum/shopping mall environments). Not only its generic design, which makes RAPOSI applicable to (m)any radio based technologies, but also RAPOSI's capability to locate passive (non-cooperative) devices (like Bluetooth/WLAN based handhelds/mobile phones) permits RAPOSI to be deployed in a quite universal manner.

Apart from highly expensive positioning hardware solutions [3],[4], existing positioning systems (and the vast majority of RSS-based systems) like RADAR [5] or EKAHAU [6] solve the uncertainty of unpredictable radio signal propagation indoors, by extensively analyzing RSS conditions at many different locations during an offline phase. This is not only time consuming, but also not reusable if anchor devices change their position or if simply the environment changes. Apart from that, these systems require a minimum of three access-points in range, which are not available in many scenarios. Furthermore, these systems are not capable of tracking devices that are not actively participating in the network.

Whereas a great amount of scientific research efforts concentrated on simulations (e.g. with OMNET), investigating many different algorithms in order to improve positioning accuracy, many other investigations concentrate on the integration of special positioning hardware, also in pursue of accuracy enhancements. However, market surveys have shown that vendors, as well as customers would only utilize location-sensing technologies, if these enabling technologies were *low cost*, *easily installable* and would *support a large scale of wireless technologies*.

Whereas we clearly forbear from integrating highly expensive positioning hardware, we adopt the benefits of sophisticated radio propagation model-based strategies as well as the advantages of light weight positioning algorithms. By means of extensive environmental radio scans, we first integrate all available radio enabled devices in the target zone (Access Points, WLAN-, Bluetooth-, RFID-devices) and if this is not enough, we cover the environment with inexpensive beacons (Bluetooth, RFID tags). This strategy allows a quick setup, especially interesting for emergency scenarios.

3 System Design

Generally spoken, it is possible to estimate the position of any radio device, as soon as the radio-signal strength of at least three non-collinear anchor devices can be measured. This is not only true for 802.11a/b/g-enabled devices, but also for Bluetooth, RFID, ZigBee or WiMax systems. It therefore makes sense to design a system, which is capable of sensing radio-enabled devices, no matter which specific transmission/reception technology is used.

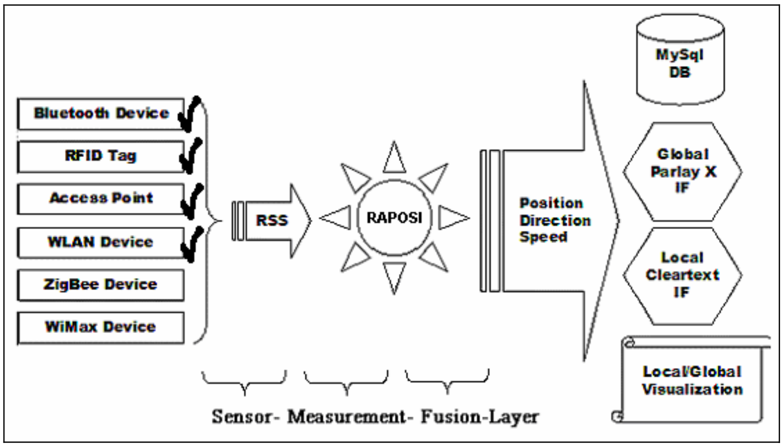


Fig. 1. Overview of hybrid RSS-based Localization of RAPOSI, Positioning Layers and Interfaces

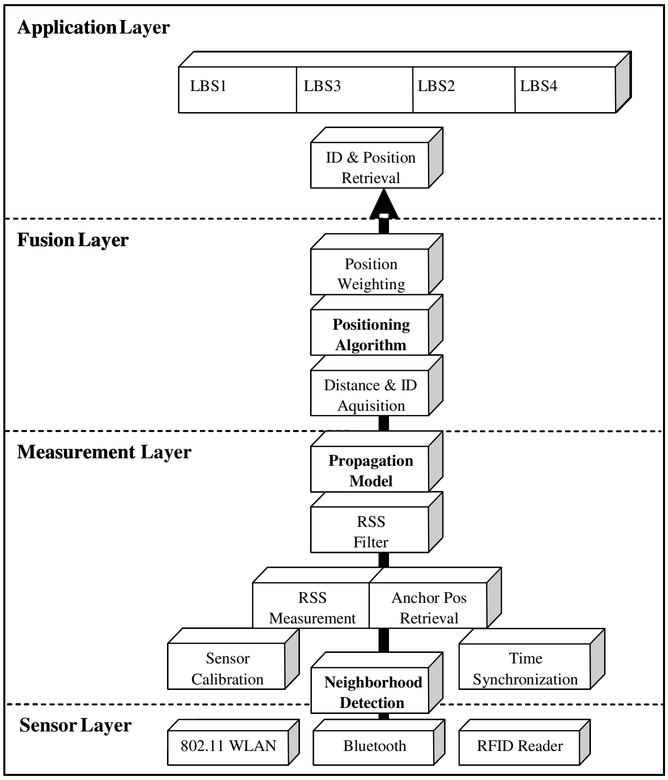


Fig. 2. Explains modules and components, required for continuous RSS-retrieval, -filtering, distance computation, position estimation and adaptive position filtering

Whereas Fig. 1 shows the different RSS measuring interfaces, as well as RAPOSI’s different interfaces for position propagation, Fig. 2 outlines the required components of each specific architecturally independent positioning layer. Whereas neighborhood detection mechanisms, as well as RSS indication and retrieval, clearly differ from technology to technology, all following, higher layered mechanisms are identical for all participating radio-based interfaces.

5 RAPOSI’s Layered Architecture

5.1 Sensor Layer

The sensor layer consists of location and/or radio signal strength sensing hardware and software drivers for detecting raw data. As recommended by currently active scientists [7] [8], a combination of many different wireless technologies improves not only location accuracy/availability, but also stabilizes the localization in case of device-/network- failures or break downs. Since our architectural premises were low cost and broad availability, we primarily focused on WLAN, Bluetooth and RFIDs.

The big drawback of WLAN and Bluetooth in terms of *neighbourhood detection* mechanisms is the necessity to set the wireless card into a *scanning mode*. While operating in scanning mode, both technologies are disconnected from the network. While with WLAN, the scanning mode has only to be triggered if Access Point scanning is desired, Bluetooth scanning is always necessary, if neighbouring IDs (MACs) have to be discovered. Furthermore, a typical WLAN Access point scan requires the card to detach from the network for at least 100-150 milliseconds, while Bluetooth scanning can take up to 10 seconds. Although conceivable, the integration of WLAN Access-Point scans always comes with the disadvantage of increased packet losses. Measurements have shown that if triggered every second, Access-Point scans result in packet loss rates of 15 to 20 %. In contrast to WLAN Access-Point scans and Bluetooth neighbourhood detection, WLAN neighbourhood detection in ad-hoc networks and RFID tag scans are highly suitable for real-time neighbourhood detection and positioning.

Table 1. The Sensor Layer output includes radio signal strength measurements as well as information about neighbouring node’s IDs and their positions

Interface	Measurements	Fixed Informaton
WLAN	Time I ID-Self I ID-Neighb. I RSS	Type I Fix→X,Y-Pos
Bluetooth	Time I ID-Self I ID-Neighb. I LQ	Type I Fix→X,Y-Pos
RFID	Time I ID-Self I ID-Neighb. I RSS Time I ID-Self I ID-Neighb. I Dist.	Type I Fix→X,Y-Pos

5.2 Measurement Layer

Different wireless technologies, deliver different metrics for perceived signal strength. Apart from radio signal strength (RSS), denoted in dBm or mW, the received signal strength indicator (RSSI) but also perceived radio signal quality or even the bit error rate (BER) are being reported. Moreover, even with one specific transmission technology, there exist big differences of transmission levels, not only due to chipset differences (e.g. Bluetooth classes I-III), but also due to differences in hardware integration (e.g. built-in vs. external). What makes conversion even more difficult is the fact that RSSI specifications differ from vendor to vendor.

Therefore, in order to derive distance from RSS measurements accurately, we integrated a generic mapping methodology. Either one provides the positioning system with a number of parameters (transmission-/reception gain, frequency) which clearly specify how to translate and convert measured RSS(I) metrics into meaningful units, or the parameters have to be specified empirically, by measuring them at fixed, known distances, with the help of already calibrated receivers and transmitters.

At the measurement layer we designed RAPOSI to deliver both classes of positioning strategies, i.e. mechanisms that continuously map RSS to distances (with the help of propagation models and positioning algorithms), as well as proximity measurements that only deliver information about whether or not a device is in range.

Extensive RSS to distance measurements of the GRIPS [2] system, analyzed the feasibility of measuring WLAN-, Bluetooth-, shot-range and long-range RFID-RSS in order to derive distance information by applying appropriate radio propagation models. The results of these investigations showed that apart from WLAN-RSS measurements, all other participating technologies performed rather weakly in the short-range (0-10m) case. With Bluetooth, no matter if Link-Quality or RSSI is being measured, high short-range RSS resolutions (i.e. exploitable RSS amplifications with increasing distances) are not achievable. Similarly, RFID short-range RSS measurements lack an exploitable trend in signal attenuation required to derive distance information. Whereas we could not find appropriate solutions in order to enhance short-range positioning resolution with Bluetooth, the incorporated RFID technology offered exploitable mechanisms for resolution enhancements.

It showed that by varying the transmission power of the RFID reader, much better short range granularities can be achieved than by measuring continuous RFID RSS only. Therefore we decided to measure the proximity of short range (0-8m) RFIDs like depicted Fig. 3, whereas we measure continuous, quantized RSS from all other long-range signal sources (WLAN, Bluetooth and Long Range RFIDs).

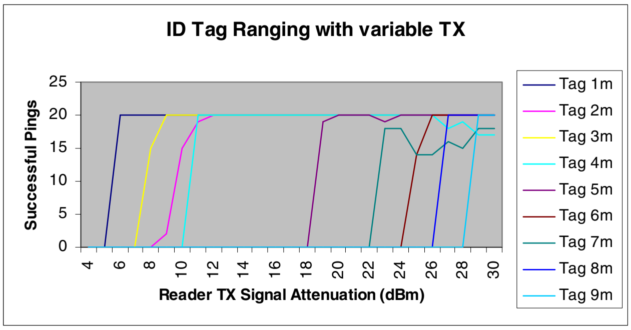


Fig. 3. By varying Transmission Powers of the RFID reader, the scanning range is set

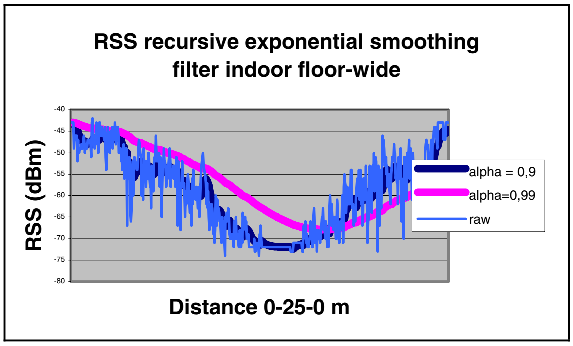


Fig. 4. Filtering Radio Signal Strength with exponential smoothing filter, showing results for two different filter intensities (alpha)

Table 2. The output of the measurement layer represents a set of distances to anchor nodes of a specific mobile node, which were computed by retrieving, filtering RSS measurements and applying appropriate propagation models.

ID	ID of mobile node to be positioned
Neighbour ID	Anchor node ID
Neighbour Pos	Anchor node's position
Distance	Distance based on propagation model
Uncertainty	Based on preliminary empirical results
Timestamp	Indicates the time when the measurements been taken

After RSS is acquired and before it is mapped to distances with the help of empirically determined propagation models, a recursive exponential smoothing filter is being applied. A filter-intensity (alpha in Fig. 4) is chosen that on the one hand appropriately filters RSS measurements, but on the other hand only adds little delay to its results.

5.3 Fusion Layer

Collaborative nodes that run the scanning, positioning and broadcasting client software are the vitally important components of the location-enabled network; in our prototype system we use 3950 IPAQs with Bluetooth-, WLAN-Interfaces and PCMCIA UHF RFID-Readers, running familiar Linux.

Whereas static evaluations of positioning algorithms and position filters of GRIPS [2] determined the proper choice of integrated components, the real-time evaluation of RAPOSI for mobile scenarios shown in Fig. 5 for the first time elucidates overall performance of the positioning system. Due to computational inconsistencies of the trilateration algorithm in times of measurement errors and due to imprecise results of stable trigonometric algorithms even in times of highly accurate distance measurements, the Mass-Spring model was selected, implemented and fine-tuned. Furthermore, the efficiency of Kalman position-filtering mechanisms was recognized and therefore implemented into RAPOSI.

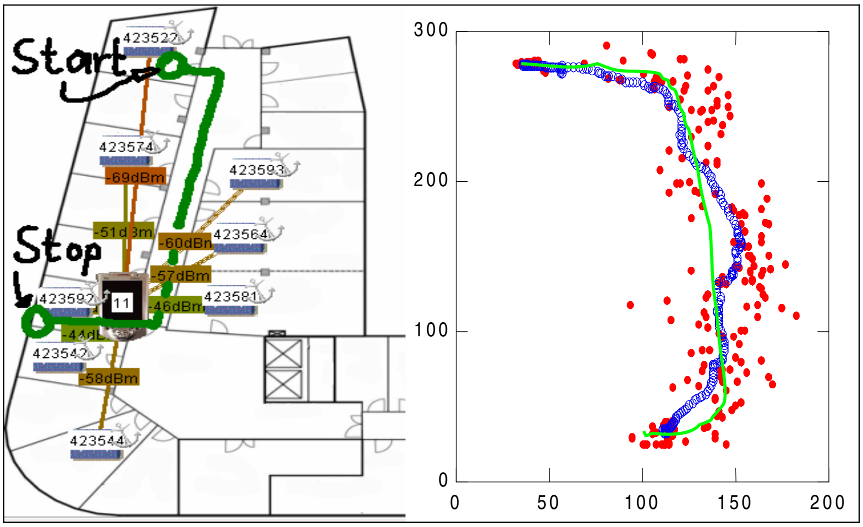
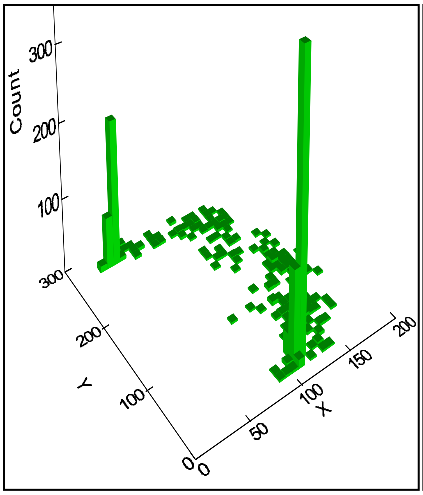


Fig. 5. The left image shows a snapshot from RAPOSI's visualization component. Node "11" is being localized while moving through the building. 8 long-range RFID tags are responsible for continuous tracking of the constantly moving mobile node "11". The right image illustrates the true track (thin), the raw (dotted) and the Kalman-filtered (strong) position estimations. Note that the origin of the coordinate system of the visualization is in the upper left corner, whereas the origin of the coordinate system of the right figure is in the bottom left corner, hence the "y"-axis has to be flipped. One unit corresponds to 10 cm.



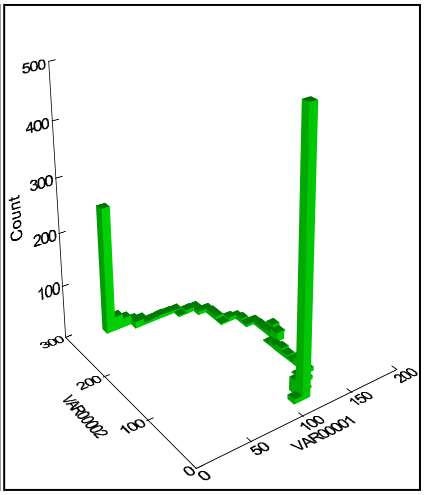


Fig. 6. Raw location estimations **Fig. 7.** Kalman-filtered location estimations

The scatter-plot in Fig. 5 (right) illustrates the raw and filtered position estimations of a moving PDA. With 8 long-range RFID tags as reference points, for a 600 sqm area, position estimations with accuracy of 2m were achieved. Fig. 5 and the densityplots of **Fig. 6** and location estimations **Fig. 7** elucidate the efficiency of the Kalman Filter, which improves position estimation by 2 to 3 orders. The two bars of high position estimation densities of **Fig. 6** and location estimations **Fig. 7** correspond to 600 and 300 time units (10 and 5 min) during which node "11" waited at the initial and at the final location respectively. Initialization took 10 seconds per RFID tag, during which the administrator placed the tags and, by pressing on the specific location of each tag on a map, wrote their position to them.

Table 3. The output of the Fusion Layer delivers location information about participating devices

Position + ID	Absolute and/or relative, obtained by superimposing and filtering all available position information
Uncertainty	Derived by means of geometrical reasoning, based on empirical results
Timestamp	Indicates the time when measurements have been taken

6 Conclusions and Outlook

Although a great number of scientific research efforts intensively dealt with indoor positioning strategies and concepts, yet there is no common agreement on necessary procedures, recommended technology or data exchange standards. Therefore, from a scientific point of view, the most valuable part of RAPOSI is the attempt to unify and classify specific procedural steps. Whereas [7] differentiates sensor-, measurement- and fusion- layer, the "location stack" proposed in [8] additionally differentiates four higher layers that deal with applications that deliver derived information like orientation, position history, acceleration and speed. Whereas we did not concentrate on the higher layers of these frameworks, we studied and implemented a generic location sensing architecture, differentiating sensor-, measurement- and fusion-layer modules. The system's overall tracking accuracy of 2 meters; the spot-on short-range RFID-tag based tagging accuracy of 50 cm, as well as its rapid installation schema makes RAPOSI suitable for many indoor scenarios.

This work has been performed in the context of FOKUS 3G beyond Testbed, known as "National Host for 3Gb Applications", sponsored by the German ministry of Education and Research (BMBF). This testbed provides different kinds of fixed and wireless next generation network technologies and related service delivery platforms. [9] [10].

References

1. T. Magedanz, F. Schreiner, H. Ziemek, "ORPS Omni Radio based Positioning System for indoor environments", Fraunhofer FOKUS, in proceedings of the Eurescom summit, Heidelberg, 2005.
2. T. Magedanz, F. Schreiner, H. Ziemek, "GRIPS Generic radio based Indoor Positioning System", Fraunhofer FOKUS, in proceedings of the 2nd workshop on positioning, navigation and communication (WPNC `05), Hannover, 2005.

3. Jeffrey Hightower, "SpotON: Ad-hoc Location Sensing using Radio Signal Strength", 2001.
4. N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, “The Cricket Location-Support System”, MIT Laboratory for Computer Science, In Proc. 6th ACM MOBICOM, Aug. 2000.
5. P. Bahl and V. N. Padmanabhan, “RADAR: An In-Building RF-Based User Location and Tracking System”, Proceedings of the IEEE Infocom 2000, Vol. 2, pp. 775-784, Tel-Aviv, Israel, 2000.
6. Ekahau, Inc., Ekahau Positioning Engine™ 2.0 Specification, http://www.ekahau.com/pdf/Ekahau_Positioning _Engine.pdf, 2003.
7. Jeffrey Hightower, Barry Brumitt, and Gaetano Borriello, “The Location Stack: A Layered Model for Location in Ubiquitous Computing,” in Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), (Callicoon, NY), pp. 22-28, June 2002.
8. Walter Lara, “Universal Location Framework: A New Wireless Building Block”, IntelDeveloperUPDATEMagazine, Feb. 2003.
9. http://www.fokus.fraunhofer.de/research/cc/ngni/activ_dir/national_host/content_eng.html
10. http://www.bmbf.de/en/

Integrating a New Mobility Service into the Jade Agent Toolkit

Peter Braun, Duc Trinh, and Ryszard Kowalczyk

Swinburne University of Technology,
Faculty of Information and Communication Technologies,
Melbourne Victoria 3122, Australia
pbraun@ict.swin.edu.au

Abstract. This paper reports on extending the Jade agent toolkit with a new mobility service. Jade implements a straightforward mobility model that is based on a pull migration strategy. This strategy is a good choice in specific environments, because only necessary classes are loaded. However, loading classes might cause a network overhead in other scenarios, in particular wide-area networks. In addition, Jade's code cache cannot distinguish between different versions of the same class, which is a drawback in long-running agent systems. We have implemented a new mobility model, named Kalong, and integrated it as a new mobility service in Jade. The paper also reports on first experiments comparing the migration performance of Jade's and Kalong's mobility service.

1 Introduction

Mobile agents have been introduced as a design paradigm for distributed applications. A mobile agent is a program that can migrate from host to host in a network of heterogeneous computer systems and fulfill a task specified by its owner. It works autonomously and communicates with other agents and host systems. During the self-initiated migration, the agent carries its code and some kind of execution state with it. What comprises the execution state depends on the underlying programming language and is, in the case of most Java toolkits for example, the serialized agent object and does not contain information about the state of the Java virtual machine. On each host they visit, mobile agents need a special software that we name *agency*, which is responsible to execute agents, provides a safe execution environment, and offers several services for agents residing on this host. A *mobile agent system* is the set of all agencies together with the agents running on these agencies as part of an agent-based application. To refer to a specific product we use the notion *agent toolkit.*

As typical for open distributed systems in general, an agent system cannot be considered to be homogeneous in the sense that all agencies are of the same type. Therefore, the community has to face the problem of interoperability between agent toolkits, that is, agents programmed for different agent toolkits (and probably different programming languages) must be able to communicate to each other and, in the case of mobile agents, to migrate between agencies of different types.

In the area of agent communication, the FIPA organization has already defined standards for agent communication languages that are widely accepted by the community and have been implemented in various toolkits. Regarding agent mobility, the current situation is still immature, although a few approaches have been proposed in the last years (see Sec.2). In particular, both standards that have been developed for agent mobility, namely MASIF [1] and FIPA (PC00087), have not been accepted by the community so far.

The reasons for this are manifold, in our opinion. First, any standard for agent mobility must be independent of a specific programming language to become accepted. However, implementing an agent toolkit that is able to execute agents developed in other programming languages, is technically hard to realize. Compare, for example, early work [2] on migrating code in different intermediate formats (including source code and binary code). In the last years, the community lost sight of the language problem, because Java became the most popular–and a *quasi* standard–programming language for mobile agent toolkits. Second, implementing a standard causes non-negligible programming effort that research groups working on single research topics (such as security, communication, development methodology, etc.) are not willing to spend. This has lead to the very disappointing situation that there are tens of different toolkits available, which are all not compatible to each other and, even worse, are sometimes only tailored to one specific research topic. For Johansen [3], this is one reason why the interest in mobile agents has dwindled over the last years. If a reference implementation of a standard had been available as independent software component, which is ready to be plugged into an existing toolkit, the chance of accepting the standard would have been higher. Third, implementing mobile agents in Java is technically very easy, because this language provides all necessary techniques, such as object serialization, dynamic class downloading, and remote method invocation as built-in functions already.

Complementing existing work in the area of interoperability between mobile agent toolkits, we propose a Java software component for agent mobility, named Kalong [4]. Kalong has been designed to be independent of any existing agent toolkit and only requires that mobile agent classes implement the Java *Serializable* interface. We have already successfully integrated Kalong into the Tracy toolkit [5] and in this paper we report on our experience integrating Kalong into the Jade toolkit [6]. We chose Jade, because it is a frequently used toolkit in the area of multi-agent systems (but not mobile agent system) and it is already used in large projects such as openNet. On the other hand, Jade's migration implementation is known to be not very efficient [7]. It turned out that the integration of Kalong as a new mobility service into Jade was straightforward without any changes to Kalong or Jade. In addition, the migration performance of Jade agents can be significantly improved using Kalong. As a consequence, it is now possible to let agents migrate between Tracy agencies and Jade agencies, which is a first step towards full interoperability of mobile agent toolkits. At the moment, we are implementing approaches from literature [8] to execute agents from foreign agent toolkits using adapter classes.

The rest of this paper is structured as follows: The following section reviews other approaches for mobile agent toolkit interoperability. In Sec. 3 we describe Jade's mobility model and in Sec. 4 we informally describe our Kalong mobility component. Then, we describe our integration approach and in Sec. 6 we compare the migration performance of Jade's current mobility technique with the new Kalong one. Finally, the last section summarizes our work and gives an outlook to further developments.

2 Related Work

We are aware of only a few approaches for mobile agent interoperability so far. An early approach for standardization in the area of mobile agent toolkits was MASIF [1], which focuses on an agent transport protocol and some common functions that every MASIF-compliant toolkit must implement. MASIF was implemented only by a few toolkits, such as Aglets [9] and Grasshopper [10]. The FIPA organization, mainly known and accepted for their standards in the area of agent communication languages, proposed a standard for agent mobility, which was withdrawn shortly after its publication. An approach that combines the FIPA communication standard with MASIF was proposed by [11].

Two approaches are known, in which new interfaces between agents and agencies were defined. For example, Magnin et al. [12] propose to program agents using a common *Guest* API, which defines functions that are available on all agencies. This API must be implemented for each toolkit and works as an adapter to the underlying agency. The paper does not describe how agent migration between different toolkits is implemented. Grimstrup et al. [13] describe a similar approach to make four toolkits interoperable by defining a standard API for toolkits and implementing two *translators* for each toolkit, namely one to translate agents developed for a specific toolkit to the new API and another one to translate from the new API to a specific agent toolkit. Agent mobility is implemented in a straightforward way using Java's built-in techniques.

Pinsdorf and Roth [8] propose to integrate mobile agent toolkits using a bottom-up approach. Their goal is to provide foreign agents (from Jade and Tracy so far) the same execution environment as in their home toolkit. Therefore, the agents' base class is emulated in their own Semoa toolkit, including packages for communication or migration, which are extracted from the source code of foreign toolkits and integrated in Semoa. The authors report on a successful communication of Jade agents running on Semoa with original Jade containers and a successful migration of a Tracy agent using the Semoa migration protocol.

3 Jade's Mobility Model

The process of agent migration is implemented in current toolkits in different ways and it is fair to say that each mobile agent toolkit implements its own *mobility model*. This notion is used in the following to describe the design decision made regarding agent migration. In order to compare existing toolkits, we have proposed a formal language to describe a toolkit's mobility model [4]. Due to the limited space, we cannot present the full description of Jade's mobility model in this paper. However, we highlight the most important issues.

From a programmer's view, Jade provides a base class for all agents, which defines a method (doMove) to initialize the migration process. After the agent has been received on the destination container, a fixed method (afterMove) is called. Agents can only migrate to containers (agencies) of the same Jade platform (a platform consists of a set of containers, comparable to an agent system), but not to containers of different Jade platforms. The address of the destination container has to be given as a logical container name, without specifying a network protocol or port number. Agent's code must be available in the local classpath and cannot be loaded from an arbitrary location.

From an agent's view, Jade uses the standard Java serialization mechanism to obtain the agent's state and implements a *pull-per-class* migration strategy to load code dynamically on demand from the agent's home container. It is not possible to define other code servers. A new network connection is opened for each class as it is not possible to combine classes to JAR files, and classes cannot be compressed or encrypted. The intuitive advantage of the pull-per-class strategy is that only *necessary* classes are loaded. A major drawback of pull-oriented migration strategies in general is that it must be possible to open network connections to the home agency, which makes this strategy inappropriate for agents that have been started on a node with unreliable or volatile network connections.

In Java, a *class loader* is responsible for loading one or many classes and every class is loaded by exactly one class loader object. Inside a class loader, classes are identified by their name, i.e. two classes bearing the same name are expected to be identical. Different class loaders must be used to create different name spaces, which has the consequence that classes loaded by different class loaders cannot *see* each other. Java's class loader implements a *code cache* that prevents loading of the same class. Actually, Jade uses the same class loader to load all agent classes from the same home container (if agents are started on different containers, the code cache has no effect). This leads to the problem that different versions of the same agent code (and from the same container) cannot be distinguished.

From the network's view, Jade uses Java RMI to communicate between mobility service instances on different containers.

4 Kalong

The Kalong mobility model is the result of in-depth analyzes of mobile agent's drawbacks compared with client-server approaches [4]. Our thesis is that mobile agents should be able to adapt and modify their migration strategy according to specific environmental parameters, such as the itinerary, the code size of each class, the probability that a class is used at the next destinations, network bandwidth and latency, the existence of code servers, etc. during run-time. With Kalong, the migration strategy is not fixed but can be *programmed* by the agent during run-time. Imagine a migration strategy as a program for a virtual machine that is responsible for agent migration. Kalong differs from current mobility models in three main aspects:

1. Kalong defines a new agent representation and new transmission units. In our model, mobile agents not only consist of an object state and their code, but have also an *external state*, which comprises of data items that are not part of the object state. Whereas the object state is still completely transmitted during an agent migration, external data items can be sent back to the agent's home agency and later be loaded from there again. A mobile agent's code is no longer transmitted in form of classes or JAR files, but we introduce a new transmission format that we call *code unit.* A code unit comprises of at least one class or many classes, which are supposed to migrate together. A single class can be member of several units.
2. Kalong defines two new agency types, additionally to the already known *home* and *remote* agencies as in current mobility models. We introduce a *code server* agency, from which an agent can download code on demand, and we introduce a *mirror* agency, which is an exact copy of an agent's home agency. A mobile agent can define an agency to be a code server or mirror and later release it again dynamically during runtime. It is important to understand that agency types are only valid for a single agent, that is, a single agency can be a mirror for one agent and a remote for another agent at the same time.
3. Kalong defines a new class cache mechanism, that not only prevents class downloading in the case of pull strategies, but also code transmission in the case of push strategies. Our class cache is able to avoid transmission of identical classes used in different agents and can distinguish between different versions of the same class.

Basic commands of the Kalong virtual machine are, for example, sending an agent's state or sending an agent's code unit (we call this *pushing* of code). Classes that were not sent during migration must be loaded (pulled) on demand later. It should be obvious that using these primitives, we can for example push all code units of an agent at once (no pulling necessary later), or do not push any code units (and pull all of them on demand). In addition, this may also result in any kind of mixture of push and pull strategies, for example, to only push the agent's main class and to dynamically load other classes. Additionally, the migration strategy also defines, which data items of the external state should be migrated to the next destination or, for example, sent back to the agent's home agency.

Kalong is available as an independent software component that only requires that mobile agent classes implement Java's *Serializable* interface. Kalong provides an interface (*IKalong*) that defines all methods to conduct a migration process, as indicated previously. If Kalong has received an agent, it informs the agency to start the agent using a second interface (*IAgentManager*) that has to be implemented by the agency.

5 Integration of Kalong into Jade

This section describes our approach for integrating Kalong as a new mobility service for Jade version 3.3.

The Jade runtime architecture contains two layers: On the platform layer, Jade can be considered as a middleware that consists of containers distributed among different hosts. On the agent layer, Jade is a set of agent objects that communicate to each other using ACL messages. Services are located on the platform layer and are, for example, those for messaging and mobility. Services are distributed between containers and on each container, there is a *slice* of that service. Slices communicate to each other using Java RMI. In order to let agents communicate to services, agents produce events using *service helper objects*. These objects translate events to *vertical commands* and pass them to the platform layer. Each command is then given to all services on that container and each service has a chance to process it (in most cases, only one service processes a command). For example, the message service processes vertical commands related to an ACL message to another agent and the mobility service processes commands related to migration events. In the case that a vertical command induces some action on a remote container, the local service translates the vertical command into a *horizontal command* that is sent to a remote slice of the same service located on a different container. The receiving slice may translate the incoming horizontal command into a vertical command again, which is then given to all services on the remote container.

To integrate a new mobility service in Jade, we had to provide a new helper class for agents to initialize the migration process. This class changes the agent's state to *TransitLifeCycle*, which causes a vertical command to be created and given to the set of all services. Inside the new service, we create an instance of the Kalong component within the service initialization phase, which includes starting a network server that accepts incoming network connections on a specific TCP port. The service has to process the vertical command and initiate a migration using Kalong. This means that we do not use the concept of horizontal commands as in Jade. For informing the main container about the transferred agent, we still use the slice concept and call method *transferIdentity* of the main slice.

The following source code shows a part of the method that is called to process the vertical command that was issued as a result of an agent migration request. The first block of statements is used to register the agent with the Kalong component and to define code units. In this example, we use a push-strategy whereas in general the migration strategy can be chosen at run-time. We define one code unit for all agent classes (i.e. the agent's class closure without ubiquitous classes that belong to Java, Jade, or the new mobility service).

The second block of statements is used to start the migration process according to the Kalong protocol. After sending a header message (containing agent name, home agency name) and the *agent definition block* (containing all class names and a MD5 hash value for each class to distinguish versions), we finally send all code units (push-strategy) and the agent's state. Note, the source code only shows an excerpt for the sake of clarity.

```
void handleInformMoved(VerticalCommand cmd) throws ServiceException
{
  // first block to register the agent with Kalong
  iKalong = kalong.getKalongInterface(agent.getName());
  iKalong.startTransaction();
  iKalong.registerAgent((Serializable)agent, agent.getName());
  String[] allClasses = iKalong.getClassNames();
  String[] filterClasses = new String[] {"java.*", "jade.*",
                                        "kalongservice.*"};
  String[] agentClasses = ArrayUtils.filter(allClasses,filterClasses);
  iKalong.defineUnit(agentClasses);     // define only one unit
  // second block to start the migration process
  Object handle = iKalong.startTransfer(destinationURL);
  iKalong.sendHeader(handle,IKalong.NOOP);
  iKalong.sendADB(handle,false);
  iKalong.sendUnits(handle,unitids);    // push the code unit
  iKalong.sendState(handle,dataItems);  // send agent's state
  iKalong.commit();
}
```

On the receiver side, Kalong receives the agent from the network and informs the mobility service about this event using interface *IAgentManager*. The service requests the container to add a new local agent and to start (powerUp) it.

6 Migration Performance Evaluation

In this section we report on first experiments that we have conducted in order to compare the performance of Jade using its built-in mobility service with Jade using our new Kalong-based mobility service.

First, we set up few experiments in our department-wide local area network (100 Mbit/sec). However, as we expected from our experience from previous performance experiments [4], it was not possible to measure any significant performance difference here. Therefore, we choose a scenario in a wide-area network, although the number of computers available for such an experiment is quite small. Our experimental set-up consists of only three computers, of which one is located in Melbourne, Australia and two computers are located in Jena, Germany. All computers are connected to a department-wide local area network (100 Mbit/sec), which is connected to the Internet via a router. The network quality of the uplink is not known. All computers run a Linux operating system (Kernel 2.6.11) and Java SDK 1.4.2. Processor speed and main memory size are not important in this experiment (in contrast to those experiments in the local area network), because the average time an agent spent on each computer is less than 10 ms, whereas a typical migration costs at least several thousands ms.

In our experiments, an agent is created in Melbourne, then migrates sequentially to both computers in Jena, and then back to Melbourne. We measure the time for the round-trip, starting right before the agent leaves Mel-

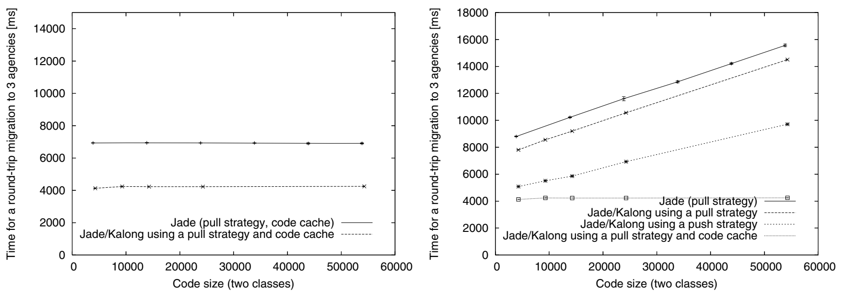


Fig. 1. Average round-trip time of an agent migrating 50 times (left) and 50 agents migrating (right) to three agencies in a wide-area network using different migration strategies

bourne and stopping immediately after the agent has been restarted in Melbourne again. Our main intention was to show the influence of different code sizes to the migration speed. Thus, each agent consists of a main class that contains the code to conduct the migration and a second class that only contains a *payload* in form of a static String variable. The size of this payload class varies between the measurements with the Jade mobility service and the Kalong mobility service, because they have been conducted by different authors. Each experiment is repeated 50 times and we only report mean values and 95 per cent confidence intervals (which are actually too small to be seen in the diagrams).

In the first experiment, the agent processes the itinerary 50 times without being restarted, compare Fig. 1 (left). In this case Jade's mobility service as well as the new Kalong mobility service are able to use the code cache. Therefore, the round-trip time is not dependent on the code size (all code is cached on each container). The figure shows that using the Kalong mobility service, migration times is about 40 per cent smaller than using Jade's mobility service.

In the second experiment, we were interested to eliminate Jade's code caching. Therefore, we re-started the container in Melbourne right after the agent returned. The new container was assigned a new container id from the main container, which had the effect that code had to be downloaded on both containers in Jena on each tour. Fig. 1 (right) shows the result for Jade's mobility service compared to Kalong's mobility service using a pull-strategy as well as a push-strategy. The push-strategy is faster than the pull-strategies because opening a new network connection for downloading each class is very expensive in wide-area networks. As we have described previously, the Kalong code cache works independent of the Java class loader concept and is even in this scenario able to cache agent's classes.

Finally, using the same set-up as in the last experiment, we compared the round-trip time for Jade's mobility service with Kalong's code-server strategy. In this case, the agent migrates with a pull-strategy but initializes a code server on the first container in Jena. When the agent has migrated to the second con-

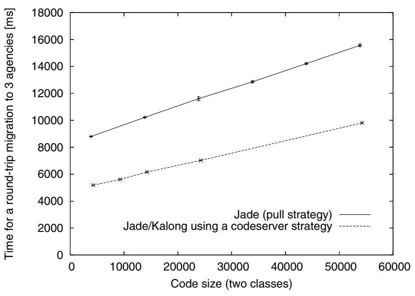


Fig. 2. Average round-trip time of a 50 agents migrating to three agencies in a wide-area network using using a pull strategy and a code-server strategy

tainer in Jena, necessary classes are then not downloaded from the home container but from the code server. After the agent has returned home, it has to send a message to release the code server again. As can be seen from Fig. 2, this strategy is even faster than a pure pull-strategy and is in this case almost as fast as the push-strategy from the last experiment. The difference is due to the time necessary to open a network connection between the two containers in Jena.

7 Conclusions and Outlook

Interoperability between mobile agent toolkits requires to enable agents to migrate between agencies of different types and, in a second step, making agents executable on a foreign toolkit. While for the second problem, several approaches from literature are known, the first problem has not been solved completely. We have proposed to use a software component that has been developed solely for the purpose of agent migration. The integration process of Kalong into Jade was straightforward and first experiments have shown that the migration performance can be increased significantly using our new mobility service. In combination with our work presented in [5], it is now possible to migrate agents between Tracy and Jade in principle. At the moment, we are working on the second problem of making Jade agents executable within a Tracy agency using an approach similar to [8].

References

1. Milojicic, D.S., Breugst, M., Busse, I., Campbell, J., Covaci, S., Friedman, B., Kosaka, K., Lange, D., Ono, K., Oshima, M., Tham, C., Virdhagriswaran, S., White, J.: MASIF: The OMG Mobile Agent System Interoperability Facility. In Rothermel, K., Hohl, F., eds.: Proceedings of the Second International Workshop on Mobile Agents (MA'98), Stuttgart (Germany), September 1998. Volume 1477 of Lecture Notes in Computer Science., Springer-Verlag (1999) 50–67

2. Knabe, F.C.: Performance-oriented implementation strategies for a mobile agent language. In Vitek, J., Tschudin, C.F., eds.: Mobile Object Systems: Towards the Programmable Internet (MOS'96), Linz (Austria), July 1996 (Selected Presentations and Invited Papers). Volume 1222 of Lecture Notes in Computer Science. Springer-Verlag (1997) 229–244
3. Johansen, D.: Mobile agents: Right concept, wrong approach (panel). In Joshi, A., Lei, H., eds.: IEEE International Conference on Mobile Data Management (MDM'04), Berkeley (USA), January 2004, IEEE Computer Society Press (2004) 300–301
4. Braun, P., Rossak, W.R.: Mobile Agents–Basic Concept, Mobility Models, and the Tracy Toolkit. Morgan Kaufmann Publishers (2005)
5. Braun, P., Müller, I., Geisenhainer, S., Schau, V., Rossak, W.R.: Agent migration as an optional service in an extendable agent toolkit architecture. In Karmouch, A., Korba, L., Madeira, E., eds.: Proceedings of the First International Workshop on Mobility Aware Technologies and Applications (MATA 2004), Florianopolis (Brazil), October 2004. Volume 3284 of Lecture Notes in Computer Science., Springer Verlag (2004) 127–136
6. Bellifimine, F., Caire, G., Poggi, A., Rimassa, G.: Jade – A White Paper. EXP in search of innovation **3** (2003) 6–19
7. Chmiel, K., Tomiak, D., Gawinecki, M., Karczmarek, P., Szymczak, M., Paprzycki, M.: Testing the efficiency of jade agent platform. In: 3rd International Symposium on Parallel and Distributed Computing (ISPDC 2004), 3rd International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogenous Networks (HeteroPar 2004), Cork (Ireland), July 2004, IEEE Computer Society Press (2004) 49–56
8. Pinsdorf, U., Roth, V.: Mobile agent interoperability patterns and practice. In: Proceedings of the 9th Annual Conference and Workshop on the Engineering of Computer-based Systems (ECBS-2002), Lund (Sweden), April 2002, IEEE Computer Society Press (2002) 238–244
9. Lange, D.B., Oshima, M.: Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley (1998)
10. Bäumer, C., Breugst, M., Choy, S., Magedanz, T.: Grasshopper — A universal agent platform based on OMG MASIF and FIPA standards. In Karmouch, A., Impey, R., eds.: Mobile Agents for Telecommunication Applications, Proceedings of the First International Workshop (MATA 1999), Ottawa (Canada), October 1999, World Scientific Pub. (1999) 1–18
11. Bellavista, P., Corradi, A., Stefanelli, C.: Corba solutions for interoperability in mobile agent environments. In: Proceedings of the International Symposium on Distributed Objects and Applications(DAO 2000), Antwerp(the Netherlands), September 2000. (2000) 283 – 292
12. Magnin, L., Pham, V.T., Dury, A., Besson, N., Thiefaine, A.: Our guest agents are welcome to your agent platforms. In: Proceedings of the 2002 ACM Symposium on Applied Computing (SAC), Madrid (Spain), March 2002, ACM Publishing (2002)
13. Grimstrup, A., Gray, R.S., Kotz, D., Breedy, M.R., Carvalho, M.M., Cowin, T.B., Chacon, D.A., Barton, J., Garrett, C., Hoffmann, M.: Toward Interoperability of Mobile-Agent Systems. In Suri, N., ed.: Proceedings of the 6th International Conference on Mobile Agents (MA 2002), Barcelona (Spain), October 2002. Volume 2535 of Lecture Notes in Computer Science., Springer-Verlag (2002) 106–120

A New Protocol for Protecting a Mobile Agent Using a Reference Clone*

Lotfi Benachenhou and Samuel Pierre*

Department of Computer Engineering, École Polytechnique de Montréal,
P.O. Box 6079, Station Centre-ville, Montréal, Québec, Canada H3C 3A7,
Phone: (514) 340-4711 ext. 4685, Fax: (514) 340-5159
samuel.pierre@polymtl.ca

Abstract. Mobile agent has several advantages compared to the traditional client/server architecture. However, a wider use of this technology is currently limited by the lack of a comprehensive security framework, especially for the protection of mobile agents from malicious hosts. This paper presents a protocol that protects a mobile agent against attacks from malicious hosts, for e-commerce applications. Using the reference execution concept, this protocol protects a mobile agent from the alteration of its code, its execution flow, its data and its itinerary.

1 Introduction

A *mobile agent* is an autonomous software component that includes program code, data and execution state. The agent programmer is called the *creator*, its user the *owner* and its execution environment the *platform*. The agent is first launched from its home platform to an initial migration platform, it then migrates from one platform to another, and finally returns to its home platform with only the pertinent data. This set of visited platforms is called the agent's *itinerary*.

The security problems in the mobile agent paradigm are twofold: *platform protection* and *mobile agent protection*. Generally, the approaches used to protect the platforms consist of authenticating and restricting the privileges of the mobile agents that visit the platform. Mobile agent protection is a highly challenging problem and a suitable solution is yet to be found. Indeed, along with its code, execution state and data, the agent is totally exposed to the platform, which can access and modify this information compromising thus its future behaviour.

The main objective of this paper is to design a security protocol that is both robust and fault-tolerant, in order to protect mobile agents from malicious attacks. More precisely, we aim to:

- Analyse the security of our protocol and evaluate its detection capacity;
- Implement two shopping agents, one secured with our protocol and the other one non-secured, for the purpose of comparisons;
- Evaluate the efficiency of our protocol and its capacity to detect attacks, through measurement and comparison to the non-secured agent.

* Corresponding author.

The paper is organized as follows. Section 2 presents a background on the threats and some related work on mobile agent security. Section 3 offers a detailed presentation of our protocol and an analysis of its capacity to detect different attacks. Section 4 presents the details of the implementation of this protocol and analyses the results. As a conclusion, Section 5 presents the synthesis of our work, its limitations as well as some indications for future research.

2 Background and Related Work

Since our main objective is to protect mobile agents, we will concentrate on the attacks a mobile agent is subject to, and address some current protection mechanisms. There are four different types of attack that a mobile agent is subject to:

- *Masquerade:* A platform can disguises itself as another platform or send the agent to another destination than the one it wants to reach.
- *Denial of Service:* A platform can ignore the mobile agent's service requests, generate unacceptable delays for critical tasks, refuse the execution of the agent or simply terminate the agent without notification.
- *Eavesdropping*: During its migration, the mobile agent is exposed to spying from platforms or other entities that can illegally access critical information or listen to the agent's communications.
- *Alteration*: When an agent arrives on a platform, it exposes its code, its state and its data to this platform, which can modify them.

The existing solutions for protecting mobile agents from these attacks are of two types: prevention on one hand and detection on the other hand.

Prevention mechanisms attempt to make it impossible or very difficult to access or modify an agent's code and state in a meaningful way. The level of prevention can be either weak or strong. An example of weak prevention is the time limited blackbox [3], which is limited by the lack of algorithms that could be used in practice. An example of strong prevention is the protection using tamper-proof devices [12], in which agents are executed in physically sealed environment. This is a very expensive solution especially for a large-scale deployment.

Detection mechanisms on the other hand aim at detecting illegal modifications of the code, the state, and the execution flow of a mobile agent. The detection can also be either weak or strong. Execution traces [10] is an example of weak detection because the size and the number of traces increase with the number of visited hosts and the detection process is only performed once the agent returns home.

In our approach, we wanted to use mechanisms that overcome these drawbacks and detect the attacks almost in real time, offering thus a strong level of detection. For this reason, we adopted the reference execution concept using a clone of the mobile agent and assumed the existence of trusted servers as explained in the following section.

3 The Protection Protocol

Using the reference execution concept, our protocol [1] ensures the protection of a mobile agent against malicious platforms attacks for e-commerce applications. To do so, in parallel to the execution of the shopping agent, its clone is executed on a trusted server in order to verify its execution. In this section, we will first present our assumptions and the entities involved in our protocol. Then, we will describe its sequences in detail, and finally we will analyse its robustness and detection capability.

3.1 Assumptions

In order to optimise traffic between the mobile agent and its clone and contain it in a certain part of the network, we assume that the network is divided into regions, and that a trusted server resides in each region. This assumption is based on ongoing research conducted on business models where the network is divided into e-supermarkets; themselves divided into e-shops [6, 9, 11]

Our second assumption is that each participating entity, namely, the buyer, the sellers' sites and the trusted servers, have a public key for encryption, a private key for signatures [7] and used in agreements with a public key infrastructure [5]. Moreover, we will use one-way hash functions in order to produce cryptographically secure compact representations of messages. In this paper, we will denote the public key of an entity *X*, E_X and its private key D_X. The hash value obtained by the application of one-way hash function *H* to a message *m* will be denoted *hm* [8].

Our third assumption is that the agent's transportation and communications are conducted via authenticated channels to prevent attacks on the mobile agent during its migration, and through the exchanged messages.

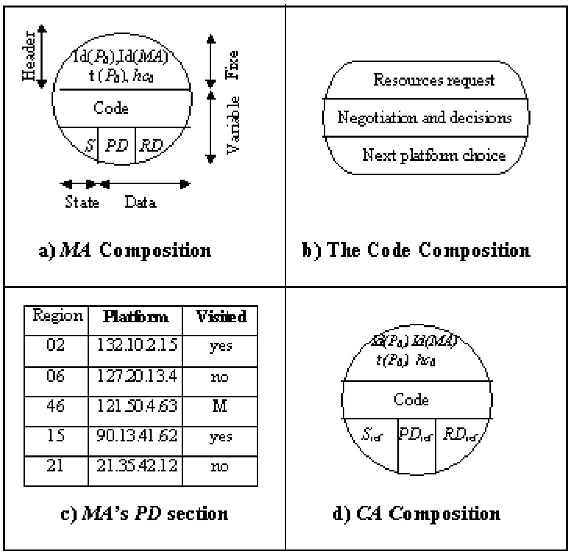
3.2 Participating Entities

There are mainly five entities involved in our protocol: the home platform (P_0), the sellers' platforms (P_{ij}), the trusted servers (TS_j), the mobile agent (*MA*) and its clone (*CA*). A brief description of each one of them follows.

The home platform, denoted P_0, creates the mobile agent *MA* and affects to it an initial itinerary, according to the product sought by the user and to the previous shopping experience. It then creates a clone, *CA*, which is an identical copy of the agent. Finally, *MA* migrates to the first platform of its itinerary P_{11}, and *CA* to the trusted server of the corresponding region, TS_1. *MA* will subsequently migrate from one platform P_{ij} to another searching the product requested by the user, whereas *CA* will migrate to another trusted server only when *MA* moves to another region. When the requested product is found, or when a stopping condition occurs, *MA* and *CA* go back to their home platform P_0 with the collected results.

In our protocol, the mobile agent is composed of four main sections, the header and the code, representing the static part, the state and the data, representing the dynamic one (see Fig. 1a). The header contains the information regarding the agent and its code, and it is in the following form: $[Id(P_0), Id(MA), t(P_0), hc_0]$. $Id(P_0)$ is the identity of the home platform and $Id(MA)$ the identity of the agent. $t(P_0)$ is a timestamp and hc_0 is the hash value of the agent code computed by P_0. The header is signed by the home platform to ensure its integrity.

The agent's code consists of the resources request, the product negotiation and the itinerary (see Fig. 1b). The third *MA*'s section is the state *S,* which contains all the information needed by the agent to continue its execution on the next platform at the same point where it stopped on the previous one. The last section of *MA* is the data section, which is composed of the protocol data denoted *PD*, and the resulting data denoted *RD*. The *PD* part contains the information of the itinerary, i.e., the previously proposed platforms, the region to which each one belongs and its status, namely, visited, not visited or malicious (*M*). This part initially contains a certain number of platforms proposed by the home platform and is subsequently updated by the visited platforms (see Fig. 1c). The *RD* part contains the resulting data, i.e., the data related to the searched product and collected on each platform, such as the price and the quality.



Region	Platform	Visited
02	132.10.2.15	yes
06	127.20.13.4	no
46	121.50.4.63	M
15	90.13.41.62	yes
21	21.35.42.12	no

Fig. 1. Composition of *MA*, *CA*, its code and the *PD* section

The clone, denoted *CA*, is an identical copy of the mobile agent *MA*. Hence, it contains the same sections and initial information. The only difference is that *CA* is only executed on trusted servers, TS_j, and, thus, the information it carries is considered as an authentic reference. This information is denoted by a suffix *ref* (see Fig. 1d). *CA* will be used to verify *MA*'s execution through a parallel execution.

The sellers' platforms, denoted P_{ij}, are the platforms that the mobile agent *MA* visits in order to find the product searched by its user. Each platform is represented by a stationary agent with whom the mobile agent and other external entities interact.

The trusted servers, denoted TS_j, are platforms that offer execution services and that the mobile agent owner trusts. As previously mentioned, in each network region, there is a trusted server whose main mission is to execute the clone *CA*, using the received information from P_{ij}, thus, validating *MA*'s execution. TS_j is also represented by a stationary agent that executes the clone and interacts with other external entities.

3.3 Protocol Specifications

This section will present the different steps of our protocol. First of all, the stationary agent of the home platform P_0 creates the mobile agent *MA* according to the product searched by the user. The composition of *MA* is thus, the header and the code, the initial state S_0 and the data; PD_0 that contains the initial itinerary and RD_0, which is empty. The clone *CA* of *MA* is then created. After that, *MA* migrates to the first platform P_1 and *CA* to the trusted server of the corresponding region, namely TS_1.

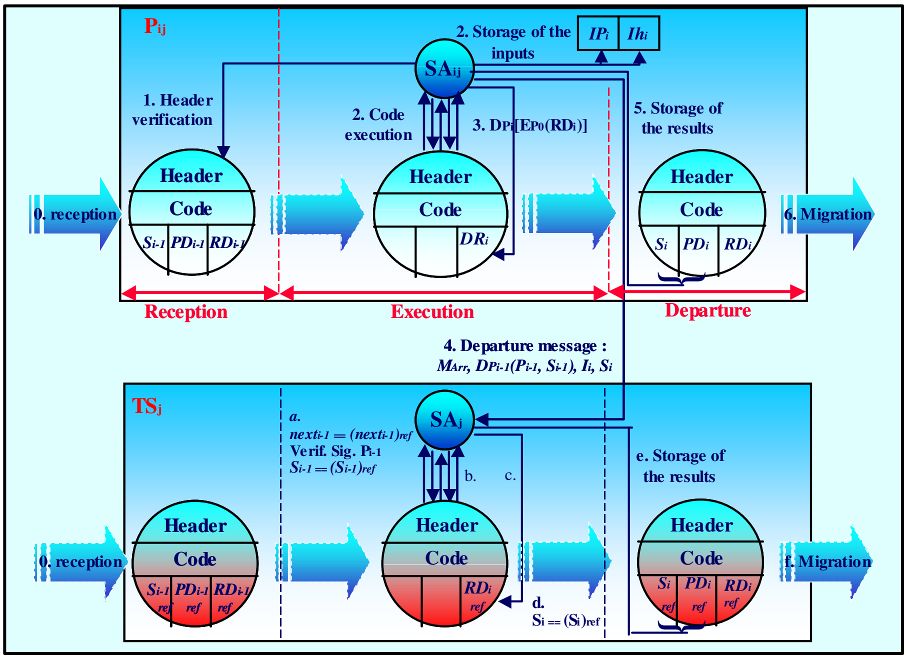


Fig. 2. Protocol steps

After this initial step, *MA* migrates according to its itinerary, from one platform P_{ij} to another while *CA* is on the trusted server of the corresponding region. The clone *CA* migrates to another trusted server only when *MA* moves to another region, and thus migrates to the trusted server of this new region. The execution of *MA* and *CA* is then performed according to the steps shown in Fig. 2.

When the mobile agent *MA* arrives on the platform P_{ij}, its stationary agent SA_{ij} checks *MA*'s certificate to confirm its identity. Then, it verifies its code integrity using the hash value hc_0 stored in the header (Step1). If the verifications generate the correct results, *MA* enters therefore in the execution stage (Step 2).

At this stage *MA* makes the request of the resources it needs (memory, CPU, etc.), executes the code corresponding to the product negotiation, then the one corresponding to the itinerary. During this execution, SA_{ij} stores all the input information used, i.e., IP_i related to the product and Ih_i concerning the itinerary. The results are then stored in the RD_i section of *MA* in the form: $D_{P_{ij}}[E_{P_0}(H(RD_{i-1}), RD_i, next_i)]$ (Step 3).

In this code section, *MA* requests recommendations from SA_{ij}, i.e., platforms where the searched product could be found. For each proposition, *MA* checks the *PD* section to verify that this proposition was not previously given by another platform. Then, the departing message $[M_{Arr}, D_{P_{i-1}}(P_{i-1}, S_{i-1}), I_i, S_i]$ is sent to the trusted server TS_j (Step 4).

M_{Arr} is the agent's arrival status that indicates that *MA* was arrived at P_{ij}, and gives the verification results. $D_{P_{i-1}}(P_{i-1}, S_{i-1})$ is the initial state signed by the previous platform, along with its identity. I_i represents all the execution information and S_i the resulting state. Finally, the itinerary information and the resulting state are stored in the agent *MA* (Step 5), which then migrates to the next platform (Step 6).

When the stationary agent of the trusted server, SA_j, receives the clone, it first checks that *MA* has migrated to the correct platform by verifying the equality $next_{i-1} == (next_{i-1})_{ref} = P_{ij}$ (Step A). Then, SA_j verifies the authenticity of the signature on the input state, $D_{P_{i-1}}(P_{i-1}, S_{i-1})$. After that, the equality $S_{i-1} == (S_{i-1})_{ref}$ is checked to verify that P_{ij} has executed *MA* with the correct input state. The clone is then executed using the received input information I_i (Step B), the results encrypted, signed and stored in the clone (Step C). Once the clone is executed, SA_j checks the validity of *MA*'s execution in P_{ij} by verifying the equality $S_i==(S_i)_{ref}$ (Step D). The resulting data and the new state are stored in the clone *CA* (Step E). Finally, if *MA*'s next migration platform is not in the same region, *CA* migrates to the corresponding trusted server (Step F); otherwise, it continues the verifications for the subsequent execution.

3.4 Protocol Security

This section addresses how our protocol can detect the previously presented attacks and designate malicious platforms. Thus, we will review all the previously presented attacks, and show which mechanisms allow their detection.

- *Masquerade:* If a visited platform P_{ij} decides to give a wrong identity to the agent, the trusted server TS_j will obtain the right one when it receives the departure message and P_{ij} will be designated as malicious. If, after completing its execution, *MA* is sent to another destination than the one it wanted, TS_j will again detect it thanks to the authentication made to receive the departure message, and the results previously found by the clone. The visited platform cannot misbehave as the correct one since it cannot counterfeit the correct private key. Finally, if a clone of *MA* is created to misbehave as the correct one, TS_j will detect it. This will be explained later as it implies the agent's alteration.
- *Denial of Service:* Since the masquerade and the alteration are addressed in other paragraphs, we will consider here that the platform sends the correct departure message. Thanks to this message and $(S_i)_{ref}$, TS_j will detect it if P_{ij} has incorrectly executed the agent's code. If P_{ij} decides to delete the agent without notification, TS_j will detect it once again, since it is constantly aware of the location of *MA*. If P_{ij} ignores the resources requested by the agent, or if it deliberately introduces delays, it will not be detected and it is very difficult to detect this kind of attacks.
- *Eavesdropping:* A platform cannot spy or modify the data collected on other platforms, as each platform encrypts and signs its resulting data. Any modification would be detected when the agent returns to its home platform.
- *Alteration:* If a platform P_{ij} modifies the agent's code by keeping the same header, the next platform will detect it thanks to the hash value and will inform TS_j. After executing the agent, P_{ij} could send wrong information in the departure message. If it sends an incorrect $D_{P_{i-1}}(P_{i-1}, S_{i-1})$, the malicious platform is either $P_{(i-1)j}$ or P_{ij} and will be detected by TS_j. If P_{ij} sends the wrong I_i and the correct resulting state, TS_j will detect the attack since the resulting state will not be consistent

with the wrong I_i. On the other hand, if P_{ij} sends wrong entries I_i and the corresponding resulting state, which is false, two cases are possible. If P_{ij} sends, subsequently, the correct state to the next platform, the alteration will then be detected and P_{ij} will be designated as malicious platform. If P_{ij} consistently sends the wrong state to the next platform, the attacking platform would be committed to sell at the suggested price and the I_i sent to TS_j will represent the proof of the negotiation. Finally, if a platform P_{ij} modifies one of the previously collected results, this will be detected and P_{ij} designated when the agent returns to P_0.

For each one of the attacks detected by our protocol, the malicious platform is designated and a recovery mechanism is triggered; the compromised agent is aborted and replaced by a copy of the clone, without losing the previously collected information.

4 Experiments, Measurements and Results

A prototype was implemented in order to assess the detection capacity of our protocol, and to measure the cost generated. We considered an application where a mobile agent moves from one platform to another, looking for the best price of a certain product. The cost criteria considered are the execution time and the generated traffic.

4.1 Experiments

The implementation was conducted using the Grasshopper mobile agents system [2], IAIK-JCE 3.01 [4] was used as security package. RSA was used for the encryption and MD5withRSA for signature with keys' length of 1,024 bits.

In addition to the implementation of a simple agent representing the non-secure application, we implemented two scenarios of our protocol. The first scenario is optimistic and assumes that all of the visited platforms are located within a same region, therefore requiring the execution of a single trusted server. The second scenario is pessimistic and assumes that all of the visited platforms are located in different regions, consequently involving a trusted server for each visited platform (see Table 1).

In order to validate our protocol, we tested the various functionalities that allow the detection of the attacks listed before during the protocol specification. For each attack, we simulated a platform behaving maliciously towards the agent or the trusted server.

Table 1. Characteristics of the test machines

Type	The platform's name		RAM (Mo)	Processor	Operating System
	Scenario 1	Scenario 2			
Desktop computer	Agency_{01}		512	P IV, 1.8 GHz	Windows 2000 pro
	Agency_{11}	Agency_{11}, TS_2			
	Agency_{21}	Agency_{12}			
	TS_1				

4.2 Measurements and Results

The performance of our protocol was evaluated for both the optimistic (*OMA*) and the pessimistic (*PMA*) scenarios and compared to a non-secured agent (*SimpleMA*). It was measured in terms of generated traffic and execution time.

Traffic analysis. For each scenario we measured the traffic generated by the mobile agent itself, the traffic generated by the clone's migration, and the one that stems from the communications (see Fig. 3).

Fig. 3 indicates that the traffic is completely generated by the mobile agent in the case of *SimpleMA*. In the optimistic scenario, the traffic generated by the clone represents almost half of the traffic generated by the mobile agent itself. This is explained by the fact that the mobile agent migrates three times whereas the clone migrates only twice, including one migration to return to the home platform, generating very little traffic on Grasshopper.

However, in the pessimistic scenario, the traffic generated by the clone is almost the same as the one generated by the mobile agent since the clone migrates as often as the mobile agent. The traffic generated by the communications is the same for the two scenarios since the same information travels through the network in both cases.

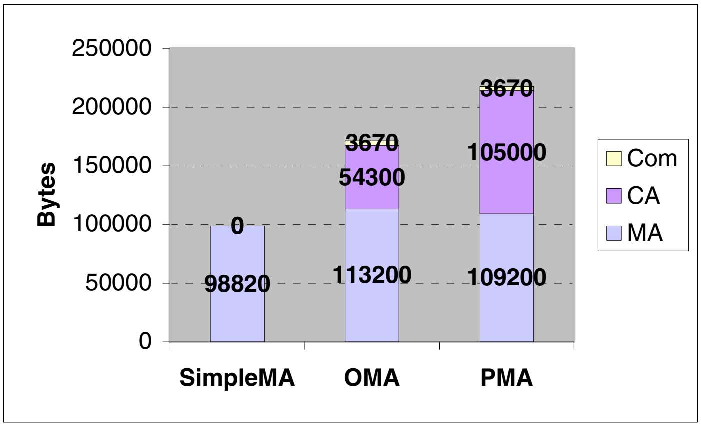


Fig. 3. Traffic generated by *MA*, *CA* and communications

Adding the security functionalities to our protocol increases traffic by 73% in the optimistic scenario and by 120% in the pessimistic one, resulting in a mean of 97%.

Execution time analysis. For each scenario, we measured the total duration of the execution of our protocol, from the departure of the agent and its clone from the home platform to the extraction of the results upon their return (see Fig. 4).

Compared to the non-secured application, the execution time increases by 47% in the optimistic scenario and by 128% in the pessimistic version, which gives an average of 87%. This quantity constitutes a small price to pay in time overhead in order to ensure the security of mobile agent applications. This overhead is due to the use of encryption and signature, the executions of the clone and the communications.

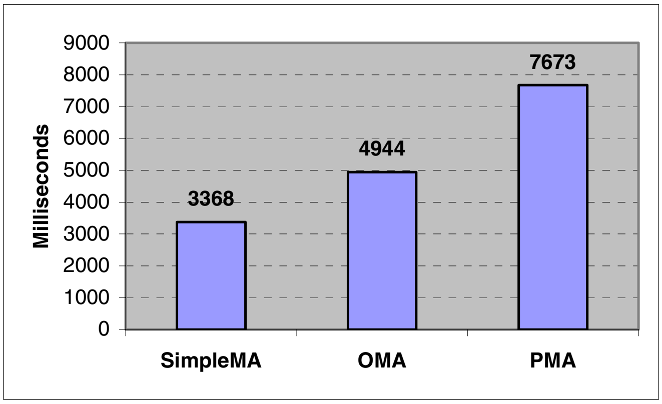


Fig. 4. Comparison of the execution times of *SimpleMA*, *OMA* and *PMA*

5 Conclusion

This paper described a security protocol that protects mobile agents from malicious platforms attacks through the use of a reference clone executed on trusted servers. An optimistic and a pessimistic scenario of our protocol were implemented and compared to a non-secured application.

Our tests showed that, in order to make a mobile agent application secure with our protocol, we have to consider an increased "price" of an average of 97% more traffic and 87% more time overhead than a non-secured application. Our protocol ensures the integrity of the agent's code, its execution, its data and its itinerary. In addition to its capacity to identify the malicious platforms, our protocol is robust, fault tolerant and does not compromise the advantages of the mobile agent paradigm.

References

1. Benachenhou, L.: Sécurité des Agents Mobiles: Protection des agents mobiles par un clone de référence, Master Thesis, École Polytechnique de Montréal, Canada, (2003)
2. Grasshopper, Release 2.2, Basics and Concepts (Revision 1.0), Online Documentation, (2001). http://www.grasshopper.de
3. Hohl, F.:Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts. In: Vigna, G. (ed.): Mobile Agents and Security, LNCS, Vol. 1419, Springer-Verlag, Berlin (1998) 92-113
4. Institute for Applied Information Processing and Communications, Javadoc for IAIK-JCE 3.01, Online Documentation (2002)
5. http://jce.iaik.tugraz.at/products/01_jce/documentation/index.php

6. ITU-T. Information Technology – Open Systems Interconnection – The directory: Authentification FrameWork. ITU-T Recommandation X.509 (1993)
7. Papaioannou, T., Edwards, J.: Mobile agent technology in support of sales order processing in the virtual enterprise. In Proceedings on 3rd IEEE/IFIP International Conference on Information Technology for Balanced Information Systems in Manufacturing, Kluwer Academic Publisher, Prague, Czech Republic, (1998) 275-288
8. Pfleeger, C. P.: Security in computing, second edition, Prentice Hall, (1996) 91-93.
9. Schneider, B.: Applied Cryptography – Protocols, Algorithms, and Source Code. In: John, C. Wiley & Sons, 2nd edition (1996)
10. Sohn, S., Yoo, K. J.: An architecture of electronic market applying mobile agent technology. In: Proceedings of 3rd IEEE Symposium on Computers and Communications, Athens, Greece (1998) 359-364
11. Vigna, G.: Protecting Mobile Agents Through Tracing. In: Proceedings of the 3rd ECOOP Workshop on Mobile Object Systems, Jyvälskylä, Finland (1997)
12. <URL: http://www.cs.ucsb.edu/~vigna/listpub.html>.
13. Vigna, G. (ed.): Mobile agent and security. Lecture Notes in Computer Science, Spring-Verlag, Vol. 1419, Berlin (1998)
14. Yee, B.S.: A sanctuary for Mobile Agents. In Vitek, J. and Jensen, C.D. (eds.): Secure Internet Programming: Security Issues for Mobile and Distributed Objects, LNCS, Vol. 1603, Berlin, Springer-Verlag (1999) 261-273
15. <URL: http://www-cse.ucsd.edu/users/bsy/pub/sanctuary.pdf>.

A Novel Approach Towards Autonomic Management in Context-Aware Communication Systems

Nancy Samaan and Ahmed Karmouch

Multimedia and Mobile Agent Research Lab.,
School of Information Technology & Engineering (SITE),
University of Ottawa, 161 Louis Pasteur St. Ottawa,
ON, Canada K1N-6N5
{nsamaan, karmouch}@site.uottawa.ca

Abstract. This paper presents a new approach for autonomic management of heterogeneous communication systems by featuring awareness of the surrounding environment's context. The contribution of this work is two-fold; the first is the development of an overall management infrastructure which identifies the necessary components for future autonomic management systems namely a context plane, an adaptive policy plane, a logic plane and an event management plane. The second contribution is the formalization of a rich model for the representation of the context of network components and services through a context hierarchy based on the utilization of the Ontology Web Language for Services (OWL-S). By describing services at different layers of abstraction semantically using a machine-processable ontology, the proposed approach allows various components to reason about the management strategies and, hence, carry-out different management tasks autonomously on behalf of administrators and users.

Keywords: Autonomic communications; context-aware; policy-based management.

1 Introduction

The control and management of large scale heterogeneous communication systems represent one of the most challenging research problems. Traditional management architectures are struggling to cope with newly emerging issues due to the ever-increasing size and complexity of the underlying network components and services. These architectures have to deal with an unprecedented amount of data that must be collected, coordinated and analyzed through the life cycle of the management operations. The management problem is further complicated by the proliferation of portable access devices (PC, laptop, PDA, etc.) along with the diversity in transmission media (Bluetooth, 802.11, etc.) and complexity of newly emerging real-time applications. With this complexity in mind, it is hard, if not impossible, for a service provider to plan a uniform resource utilization and management in all networks and to envision fixed bandwidth requirement from all users. It is even harder to project what users will demand due to their mobility and the variations in the characteristics of the terminals they may carry.

Traditional management frameworks are characterized with static management decision operations that lack the ability for situation and context adaptability. While these systems are best optimized for particular settings, they may not be suitable for continuously changing mobility scenarios. Furthermore, current management tools lack the ability to define network services and operations as a function of how business objectives and users and applications requirements change. For example, a WLAN installed in an airport lounge designed for an average level of traffic utilization is faced with either complete waste of network resources when experiencing periods of modest communication traffic and may completely fail when experiencing periods of dense communication traffic due to flight delays as a result of weather conditions.

It has become evident that one of the main requirements for a successful management architecture is the ability to be dynamic and self-reconfigurable in response to the evolving surrounding environment. Autonomic Computing, launched by IBM in 2001 [1], is an emerging technology that aims at allowing users to transit transparently and dynamically between different providers and service domains. In order to gracefully perform this transition, automated selection of service configurations, reallocation and monitoring must be carried-out with minor intervention of users and system administrators, the system must be able to make service selection and adaptation decisions autonomously and intuitively, without the users' awareness. Nevertheless, these decisions must be frequently evaluated to ensure their feasibility in a dynamic environment with continuously changing parameters. Although, in theory, autonomic computing essence seems to provide the ultimate solution for the complex management problem, in general, research efforts towards autonomic management are still in their infancy and are still faced with many challenges before being realized as a successful solution.

In previous work, the authors presented an approach for adaptive management based on the utilization of policies [2]. Based on higher-level policies supplied by network administrators and users, network-level policies were autonomically created and modified to dynamically control the behavior of various network components. In this paper, the authors further extend this approach towards the development of a novel autonomic, self-aware adaptive management system. The proposed architecture bases network adaptation decisions on information obtained from the surrounding environment context with minimal intervention from human operators and users. Through the development of a novel management framework, the authors illustrate that autonomic management is feasible in context-aware environments. The key idea in this approach, is incorporating available user, application, network and surrounding environment context into the dynamic management and decision-making process. Context information collected from different sources, such as the user schedule, user profile, terminal, and network monitors, along with tools from knowledge processing and inference reasoning are used to replace the human operators involvement in the management process and hence to provide autonomic management of future communication systems. The contribution of this work is two-fold. The first contribution is a formalization of a rich model for the context of network components and services through a context hierarchy based on the utilization of the Ontology Web Language for Services(OWL-S) [3]. The second contribution is the development of an overall management infrastructure which incorporates the necessary planes for the automated management process, namely, the context plane, the policy plane, the logic plane and the event management plane.

The rest of the paper is organized as follows. In section 2, related work and existing management approaches are discussed. Section 3 provides a brief introduction of the necessary background for the utilized ontologies. Section 4 describes the proposed automated management infrastructure. Finally, Section 5 concludes the paper.

2 Related Work

Network management is mainly concerned with the control and provisioning of network resources and services to meet operators' business objectives along with the consumers' needs. In early developed management systems, administrators have always opted to utilizing simple command line interfaces (CLI), such as telnet and Secure Shell (SSH), for direct configuration and management of network components. CLIs are heavily dependent on human operators, and usually contain vendor and even device dependent commands. They also lack a common data model, and that, in turn, would make the automation of the management process impossible. The Simple Network Management Protocol (SNMP) [4] is an application layer protocol that is used to exchange management information between network devices. It has been widely used during the early deployments of IP systems, but as complex Quality of Service (QoS) models have evolved, e.g., integrated services and differentiated services models, SNMP was faced with large data sets and tables and fell short to meet the newly added requirements of continues modification of network components by network administrators.

The Common Open Policy Server (COPS) [5] uses a different network management model where it is envisaged that the device's managed configuration will undergo continual incremental change in response to user's individual requirements. To achieve this incremental update, the network management system uses a local copy of a configuration state for each managed device, and incremental updates are sent to each managed device whenever the configuration state is altered. This introduces a new concept into the management architecture, that of a synchronized state between the management station and the managed device. While COPS was a first step towards providing a standard common model for configuring network components, similar to SNMP, COPS was based on a data-centric approach which does not contribute to adding any intelligence to the management process. The Common Information Model (CIM) [6], developed by the Distributed Management Task Force (DMTF), follows an object-orientated approach with full support of methods on data objects, which are not available in SNMP or COPS. CIM has been extended in cooperation with the DMTF to describe high-level policies as rules sets (PCIM). In general, CIM is basically limited to defining the behavior of a managed entity, e.g., a router, at a particular point in time. Furthermore, there is still a gap between management systems utilizing CIM and actual network devices.

Although, different information models have already been used to represent and configure different managed entities, thus far, these models suffer from a lack of a smart representation and encoding of available knowledge in a way which can be used to make the necessary inferences to automate the management process.

Policy-based network management has been introduced as a promising solution to the problem of network management. In current systems, policies are defined by users, administrators, or operators. Once defined, these policies are translated and stored in a policy repository. Policies are then retrieved and enforced as needed. Despite the recent research advances in the area of policy-based network management, e.g., [7, 8, 9], existing policy frameworks are mainly static in their configuration nature. In recent work, the authors have presented a novel approach for network level adaptation in response to changing higher level policies based on the utilization of forecasting functions [2].

While adaptive policies can represent one facet of the solution to the autonomic management problem, namely the adaptation process, that can guide the behavior of components and services, a smart information model must be introduced to provide guidance of *how, when* and *where* adaptation should be performed. In this case, policies become the tools that would guide the adaptation process in response to the context supplied by the information model and the inferences made based on this information.

3 Overview of the Ontology Web Language for Services (OWL-S)

Ontologies are one of the main approaches used in the scope of Knowledge Management (KM) and Artificial Intelligence (AI) to solve questions related to semantics, with current relevance in the Semantic Web. They describe an abstract model of a domain by defining a set of concepts, their taxonomy, interrelation and the rules that govern these concepts in a way that can be interpreted by machines. The main difference between ontologies and other information models common to network management, such as COPS and SNMP, is that the latter models do not include axioms or constraints, present in ontologies, which makes difficult to reason with them, and hence they do not provide the necessary tools required for automating the management process.

This section briefly reviews OWL [10], the standard web ontology language proposed by W3C recently, and OWL-S, OWL-based Web service ontology [3]. OWL is intended to provide computer-interpretable descriptions of services in an unambiguous computer-interpretable form in order to automate the processes of service allocation, selection, composition and monitoring. As shown in Figure 1 [3], the structure of the OWL-S ontology is comprised of three main subontologies, namely, The *Service Profile*, the *Service Model*, and the *Service Grounding*. A *Service* is the upper level ontology which provides the generic Service class. The service profile provides means for describing what a service does. In turn, it facilitates the process of service advertising and selection. The profile class of the OWL-S ontology specifies web service descriptions based on their functional and nonfunctional parameters. The profile specifies the inputs that a service requires, the outputs that it generates, the preconditions that must hold in order to execute the web service and the effects that an execution generates. The results may depend on the execution because of potential conditional statements. The service modelling represents a detailed description of a service's operation. While the service profile and service modelling serve as abstract specifications of web service characteristics, the service grounding enables communication with a concrete web service by binding abstract input/outputs to concrete message formats. The service grounding provides details of how to inter-operate with or access a service. One of the main advantages of the OWL-S model is that, despite the fact that it was originally intended for web services, it is very general and can be adapted into describing any type of service.

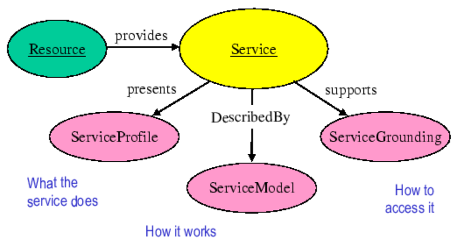


Fig. 1. Top level View of the Service ontology [3]

4 Proposed Architecture

The basic premise behind the proposed work is that, in order to realize decision autonomy, firstly, the system must possess detailed understanding of the context of the surrounding environment, which includes information pertaining to users (e.g., users' locations, preferences), terminal characteristics (e.g., battery level, CPU power, memory constraints), application service requirements (e.g., frame rates, media types) and underlying network characteristics (e.g., types of routers, transmission media, services offerings). This information must be presented in a manner that will allow reasoning and inferences concerning management functionalities. Secondly, a strong logic and reasoning component must be available to replace human operators in the decision making and adaptation processes. Finally, the system must provide a flexible means for communicating the decided adaptation operations to system components where they are applied.

Based on the the above premise, the proposed framework realizes autonomic management in four orthogonal and canonical planes as shown in Figure 2, namely, the logic, context, policies, and events planes. The logic plane represents the core compo-

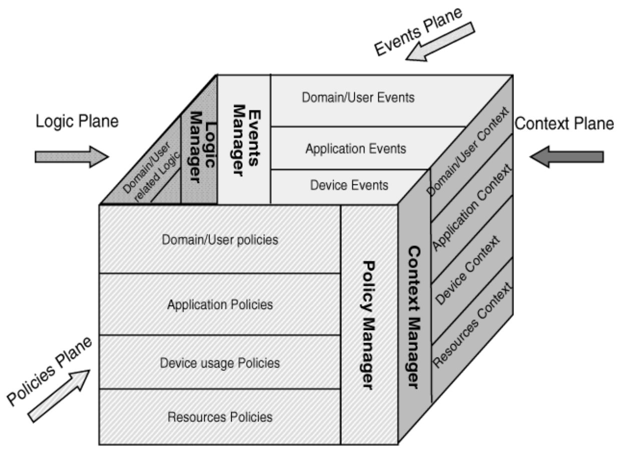


Fig. 2. Schematic description of the proposed architecture

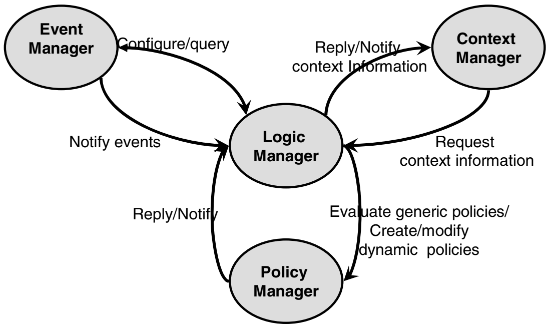

Event Manager
Configure/query
Reply/Notify context Information
Context Manager
Notify events
Logic Manager
Request context information
Reply/Notify
Evaluate generic policies/ Create/modify dynamic policies
Policy Manager

Fig. 3. Interactions and messages flow between managers at different planes

nent which orchestrates the interactions among the other planes. In functional terms, each of the four planes is composed of a number of layers and a plane manager. Figure 3, depicts the interactions between the managers of different planes. A general overview of the functionalities of different planes will be described in the following sections.

4.1 Context Management Plane

The *Context Plane* manages the persistent repository of context information. As shown in Figure 4, the Context Manager (CM) is mainly responsible for managing two different types of context information, namely, the context of the entities in the managed domain, and the context of the services offered within that domain. The former context includes the context of the participating entities in the domain spread over four layers. The first layer includes the user context, such as his location, time, preferences, along with context of the managed domain. For example, for a WLAN in an airport, this context will include location of the servers, types of services offered by these servers such as on-demand news and movies, and hotspot locations. The second layer, contains running applications' context such as available software, types of media used (e.g., voice and video) and types of configuration parameters used by each application. The third layer stores the context related to the domain devices, this includes the characteristics of the user terminals along with the domain servers. Finally, the fourth layer stores network related context such as routers capacities, configuration parameters, performance monitoring components, etc. Context updates across the four layers are obtained from various sources such as sensors and software probes. This part of the context can be modelled using any of the available models in literature e.g., [11]. The CM is also responsible for detecting context changes and reporting notifications about the changes to the corresponding layers at the event plane.

The latter type of context, that is the context of the offered services, provides the essence for management automation. For example, the services offered at the network level are services offered by edge routers to connected hosts and routers of neighboring domains or are offered by core routers to flows. At the application level services can be an email service, a web browsing service, or real-time video conferencing, etc. As depicted in Figure 4, these services are spread across four different layers. To provide

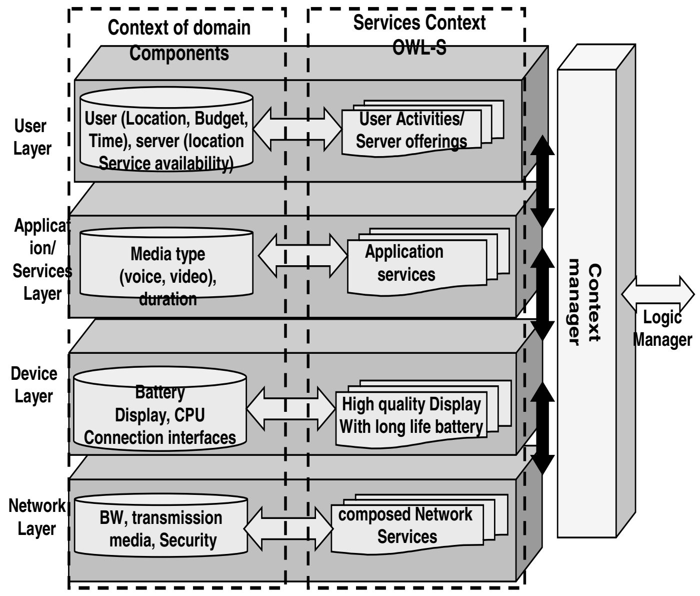

Context of domain Components
Services Context OWL-S
User Layer
User (Location, Budget, Time), server (location Service availability)
User Activities/ Server offerings
Application/ Services Layer
Media type (voice, video), duration
Application services
Device Layer
Battery Display, CPU Connection interfaces
High quality Display With long life battery
Network Layer
BW, transmission media, Security
composed Network Services
Context manager
Logic Manager

Fig. 4. Schematic description of the context management plane

a uniform description of services, OWL-S [3] is used. Each layer i provides a set of available services to the upper layer $i + 1$ through the used OWL service profiles. On the other, hand it provides the means for implementing this service by the lower layer $i - 1$ through the OWL service modelling and grounding. Figures 5 and 6 illustrate some examples of the services'context at the application and network layers, respectively.

4.2 Event Management Plane

To achieve scalability in the proposed infrastructure, instead of having the Context Manager (CM) report all context changes to the Logic Manager (LM), events that represent significant effects on the management tasks are registered by the (LM) with the Event Manager (EM). The EM is then responsible for reporting back to the LM once any of these events is triggered. In the context of the proposed infrastructure, an event is related to a change in the context of one of the entities in the managed domain or a violation of one of the offered services. Each layer of the *Event Plane* is responsible for the maintenance and the fine grained monitoring of different registered events related to the users, applications, devices and services, and network resources. Examples of adaptation-triggering events that coincide with changes of entities include the detection of changes in terminal-and/or network resource, admission of a new flow, the termination of an old flow, radio propagation degradation, handovers, etc.

4.3 Policy Management Plane

The policy plane contains the various management policies that control the behavior of the managed domain (e.g., user preferences policies, authentication/authorization

```
<!-- Service description -->
<service:Service rdf:ID ="Service_High_Qlty_Dsply">
   <!--Reference to the related profile  -- >
    <service: presents rdf:resource ="Profile_High_Qlty_Dsply"/>
     <!--Reference to the related Process Model  -- >
     <service:describedBy rdf:resource = "Hih_QLty_DsplyProcess"/>
     <!--Reference to the related Grounding -- >
     <service:supports rdf:resource = "Grounding_High_Qlty_Dsply"/>
< /service:Service>
      <!-- Profile description -->
<profile:Profile rdf:ID=" Profile_High_Qlty_Dsply ">
     <service:isPresentedBy rdf:resource="# Service_High_Qlty_Dsply "/>
     <profile:serviceName xml:lang="en"> High Quality Display
     </profile:serviceName>
     <profile:textDescription xml:lang="en">This service runs a selected
                        movie with high quality
      </profile:textDescription>
   <profile:hasInput rdf:resource="#MovieName"/>
    <profile:hasInput rdf:resource="#Framerate"/>
   ......
     <profile:hasOutput rdf:resource="#RequestedBW"/>
     <profile:hasOutput rdf:resource="#RequestedCPU"/>
    <profile:hasOutput rdf:resource="#RequestedBuffer"/>
</profile:Profile>
<!– Process Modelling description -->
<process:AtomicProcess rdf:ID=" Hih_QLty_DsplyProcess ">
                     <service:describes rdf:resource="# Service_High_Qlty_Dsply "/>
                     <process:hasInput rdf:resource="# Framerate "/>
                       ....
                     <process:hasOutput rdf:resource="# RequestedBW "/>
</process:AtomicProcess>
<!-- Grounding description -->
<grounding:AppGrounding rdf:ID=" Grounding_High_Qlty_Dsply ">
                     <service:supportedBy rdf:resource="# Service_High_Qlty_Dsply "/>
                     <grounding:hasAtomicProcessGrounding rdf:resource="#
Grounding_Process_High_Qlty_Dsply "/>
</grounding:AppGrounding>
```

Fig. 5. An example of OWL-S representation of services offered at the Application level

policies, billing policies, admission control policies). The Policy manager (PM) is responsible for creating, maintaining and adapting four types of policies, namely, the user, application, device and resource polices. This is achieved through a hierarchal policy model. At each layer, two types of policies are maintained static policies, that are configured once by system administrators or users, and dynamically created policies that are created due to requests from the corresponding layers at the logic plane. As described in details in pervious work, [2], based on the supplied objectives by the corresponding layers in the logic plane, the PM selects one or more of the policy actions required to satisfy the imposed objectives and dynamically assembles and dispatches the corresponding network-level policies.

4.4 Logic Management Plane

The Logic Manager (LM) at the *Logic Plane* is brain of the proposed infrastructure. It instruments the multi-dimensional adaptation process and provides the orchestration between the managers at the other planes. It reasons about changes in the current context, received from the event plane, and decides whether or not to perform any required adaptations. The following points summarize the main functionalities of the LM:

– It is responsible for the evaluation of triggered events received from the event logic.

```
<!-- Service description -->
<service:Service rdf:ID =“Service_Admission_Control”>
   <!--Reference to the related profile  -- >
    <service: presents rdf:resource =“Profile_Admission_Control”/>
     <!--Reference to the related Process Model  -- >
     <service:describedBy rdf:resource = “Admission_Control_ProcessModel”/>
     <!--Reference to the related Grounding -- >
     <service:supports rdf:resource = “Grounding_Admission_Conrtol”/>
< /service:Service>
      <!-- Profile description -->
<profile:Profile rdf:ID=" Profile_Admission_Control ">
     <service:isPresentedBy rdf:resource="# Service_Admission_Control "/>
     <profile:serviceName xml:lang="en">  Admission control service
</profile:serviceName>
     <profile:hasInput rdf:resource="#Host_Addr"/>
    <profile:hasInput rdf:resource="#RequestedBW"/>
   ……
     <profile:hasOutput rdf:resource="#Admitted_BW"/>
     <profile:hasOutput rdf:resource="#DHCP_Marking"/>
     …..
</profile:Profile>
<!– Process Modelling description -->
<process:ProcessModel rdf:ID=" Admission_Control_ProcessModel ">
                    <service:describes rdf:resource="# Service_Admission_Control
"/>
               <process:hasProcess rdf:resource="#
AdmissionControlProcess"/>
</process:ProcessModel>
<process:CompositeProcess rdf:ID=" AdmissionControlProcess ">
        <process:hasInput rdf:resource="#Host_Addr"/>
        <process:hasInput rdf:resource="#RequestedBW"/>
          …
        <process:hasOutput rdf:resource="#Admitted_BW"/>
           …
         <process:composedOf>
              <process:Sequence>
                     <process:components rdf:parseType="Collection">
                       <process:AtomicProcess rdf:about="#TestSLAProcess“>
                          <process:AtomicProcess rdf:about="I#TestAdmission"/>
                       <process:AtomicProcess rdf:about="I#AddMeter"/>
                          <process:AtomicProcess rdf:about="I#AddMarker"/>
                   </process:components>
                 </process:Sequence>
         </process:composedOf>
   …..
</process:CompositeProcess>
<!-- Grounding description -->
<grounding:NWGrounding rdf:ID=" Grounding_Admission_Conrtol ">
                      <service:supportedBy rdf:resource="#
Service_Admission_Control "/>
                      <grounding:hasModelProcessGrounding rdf:resource="#
Grounding_Process_Admission_Control"/>
</grounding:NWGrounding>
```

a) service example at an edge router

```
<!-- Service description -->
<service:Service rdf:ID =“Service_Gold_Treatment”>
   <!--Reference to the related profile  -- >
    <service: presents rdf:resource =“Profile_Gold_Treatment”/>
     <!--Reference to the related Process Model  -- >
     <service:describedBy rdf:resource = “Gold_Treatment_ProcessModel”/>
     <!--Reference to the related Grounding -- >
     <service:supports rdf:resource = “Grounding_Gold_Treatment”/>
< /service:Service>
      <!-- Profile description -->
<profile:Profile rdf:ID=" Profile_Gold_Treatment">
     <service:isPresentedBy rdf:resource="# Service_Gold_Treatment"/>
     <profile:serviceName xml:lang="en"> Gold class service treatment at core
router
</profile:serviceName>
    <profile:hasInput rdf:resource="#DHCP"/>
    <profile:hasInput rdf:resource="#ServiceBW"/>
   ……
</profile:Profile>
<!– Process Modelling description -->
<process:ProcessModel rdf:ID="Gold_Treatment_ProcessModel ">
                    <service:describes rdf:resource="# Service_Gold_Treatment"/>
                 <process:hasProcess rdf:resource="# Gold_TreatmentProcess"/>
</process:ProcessModel>
<process:CompositeProcess rdf:ID=" Gold_TreatmentProcess ">
        <process:hasInput rdf:resource=“DHCP#"/>
        <process:hasInput rdf:resource="#ServiceBW"/>
          …


<process:composedOf>
               <process:Sequence>
                     <process:components rdf:parseType="Collection">
                       <process:AtomicProcess rdf:about="#Test classRate“>
                         <process:AtomicProcess rdf:about="I#AddMeter"/>
                        <process:AtomicProcessrdf:about="I#RandomDroper"/>
                   <process:AtomicProcess rdf:about="I#configureScheduler"/>
                   </process:components>
               </process:Sequence>
         </process:composedOf>
   …..
</process:CompositeProcess>
<!-- Grounding description -->
<grounding:NWGrounding rdf:ID=" Grounding_Gold_Treatment">
                      <service:supportedBy rdf:resource="#
Service_Admission_Control "/>
                      <grounding:hasModelProcessGrounding rdf:resource="#
Grounding_Process_Gold_Treatment"/>
</grounding:NWGrounding>
```

b) service example at a core router

Fig. 6. An example of OWL-S representation of services offered at the network level

– It communicates with context manager at corresponding layers to obtain information about network services, and learns the users' preferences as well as causes of service violation in case of a violation event, etc.
– Based on the received events along with the obtained context, it is responsible for taking the appropriate measures to perform any required adaptation steps in response to the triggered event. Adaptations performed by the logic plane fall under one of two categories. The first are adaptations in the types of the services offered by the managed domain, this can be analogous to the adaptation performed manually in current systems by network administrators to the network information model such as CIM documents. These are communicated to the corresponding layers at the context plane. The second type of adaptations are concerned with the rules governing the offering of these services which are communicated to the policy plane. This step is analogous to manual configuration of network policies.

5 Conclusions and Future Work

In this paper we presented a novel framework for autonomic management in context-aware heterogeneous systems. The proposed work utilized concepts of OWL-S in order to provide the necessary formalization that will allow the automated reasoning and inferences concerning various management functionalities. In future work, we plan to evaluate our work through the implementation of the proposed architecture.

Acknowledgment

This work was funded by an NSERC grant.

References

1. J. Kephart and D. Chess, "The Vision of Autonomic Computing", *Computer Magazine, IEEE*, vol. 36, n. 1, pp. 41–50, Jan. 2003.
2. N. Samaan and A. Karmouch, "An Automated Policy-Based Management Framework for Wired/Wireless Differentiated Communication Systems", *accepted for publication in special issue of IEEE Journal on Selected Areas in Communications (JSAC) on Autonomic Communication systems.*, 4th Quarter 2005.
3. The DAML Services Coalition, "OWL-S: Semantic Markup for Web services", DAML, http://www.daml.org/services/daml-s/0.9/damls.html, 27th of December 2003.
4. William Stallings, *SNMP, SNMPv2, and CMIP: The Practical Guide to Network Management Standards*, Addison-Wesley, 1993.
5. D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan and A. Sastry, "The COPS (Common Open Policy Service) Protocol", in *IETF RFC 2748*, Jan. 2000.
6. Distributed Management Task Force (DMTF), "Common information model (CIM) specification version 2.2", 1999.
7. G. Valrie, D. Sandrine, K. Brigitte, D. Gladys and H. Eric, "Policy-Based Quality of Service and Security Management for Multimedia Services on IP networks in the RTIPA project", in *MMNS 2002, Santa Barbara, USA*, Oct. 2002.
8. P. Flegkas, P. Trimintzios and G. Pavlou, "A Policy-Based Qualifty of Service Management System for IP DiffServ Networks", *IEEE Network*, pp. 50–56, Mar./Apr. 2002.
9. L. Lymberopoulos, E. Lupu and M. Sloman, "QoS Policy Specification A Mapping from Ponder to the IETF Policy Information Model", in *ENC01*, Sept. 2001.
10. D. L. McGuinness and F. V. Harmelen, "OWL Web Ontology Language Overview,W3C", http://www.w3.org/TR/owl-features/, 2004.
11. M. Khedr and A. Karmouch, "ACAI: Agent-Based Context-aware Infrastructure for Spontaneous Applications", *Network and Computer Applications*, vol. 28, n. 1, pp. 19–44, Jan. 2005.

Abstraction for Privacy in Context-Aware Environments*

Eun-Sun Cho[1] , Kang-Woo Lee[2], and Manpyo Hong[3]

[1] Department of Computer Science, Chungbuk National University,
12 Gaesin-dong, Cheongju, 361-763, South Korea
eschough@cbnu.ac.kr
[2] Electronics and Telecommunications Research Institute,
161 Gajung-dong, Daejeon, 305-700, South Korea
kwlee@etri.re.kr
[3] Department of Information and Computer Engineering, Ajou University,
San 5, Wonchun-dong, Yeongtong-gu, Suwon, 443-749, South Korea
mphong@ajou.ac.kr

Abstract. Privacy protection operation may block customized services in context aware systems. Adopting good privacy policies will be helpful to avoid such unfortunate situations, but traditional access control policies are not enough for this purpose. This paper proposes a privacy policy description mechanism for context-aware systems. It provides the facilities to describe abstraction on context data, in order to prevent invading user's privacy, while keeping the context-aware serves continuing.

1 Introduction

Context is a subset of information around a user, and gathered for context-aware computing. For customized and intelligent services, services and the helper modules works with personal data collected by his/her RFID tags, RFID cards, PDAs and mobile phones.

However, not only with the bright side, service providers, marketers and others are able to see private information on a person which she/he may not want to make revealed. Steps (1) ~ (4) in Fig. 1 show examples of vulnerable points in a simple architecture for context-aware systems, which the personal information can be easily revealed. The information on RFID card that a person is carrying is read by Antenna (1), sent to the reader module (2) and DB server (3). The DB data can be used to offer POS promotions discounts, coupons, etc. for more personalized services (4). On each delivery route, the data being transferred can be seen by unauthorized subjects easily.

From (1) to (3), encryption technologies on wireless network data will work to protect privacy. In this paper we concentrate on (4) and what happens among the components of personalized services. In these cases, the possible solutions include policy based access controls.

* This research is supported by the Ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

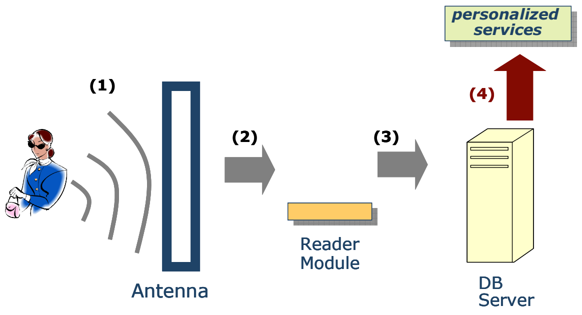


Fig. 1. An example of the flow of personal data

This paper proposes a privacy policy description language for context-aware systems and its application framework. It provides the facilities to describe abstraction on context data, which will be useful to keep context-aware serves continuing without invading user's privacy.

The sequence of this paper is as follows; the next section covers an overview on an application framework for our privacy preserving mechanism. Section 3 describes the model to define abstract views. Section 4 introduces the policy language suit with abstraction facilities, and its relationships with the application framework. Section 5 provides related works and discussions. Section 6 concludes this paper.

2 Overview of the Application Framework

According to whether source codes are given or not, different kinds of approaches are taken to defend. The main role of DB server is to manage users' data delivered via a reader module. When an application service requests context data of a user, it also sends a specific token to identify the user, like the RFID card number and the mobile phone number. DB server transformed the token into the user's ID by the help of the authentication server. Then the DB server extracts and sends back the requested context data to the application service module, as long as the policy allows the access.

Note that there can be more than one software-components intervened in the flow route. Let us call those components 'low-level data processing (services)'. Application service programmers may make use of the components directly or indirectly as useful utilities for context-aware programming.

Such software components, DB server and application services are aware of some of three categories of context-data; pseudo-user-ID's, user-ID's and user-attributes. All modules know some related user-attributes to give services. Basically DB server knows all of the three kinds of data. Authentication server knows both pseudo-user-ID's and user-id's. Application services perceive only pseudo-user-ID and a subset of user-attributes for context-aware services. Low-level data processing services may or may not know user-ID's.

The goal of the framework is decoupling the set of context data usage in order to prevent unintentional data leak while keeping high quality context-aware services continuing. To achieve this, we propose processing data according to a few principles.

The first one is to minimize commonly shared context data. Similar to decoupling principles in the software engineering area, it propose that the modules that access one set of context data are decoupled from the modules that access other set of context data. For example, *Know-user-ID-modules* are decoupled from *Not-know-user-ID modules*.

Actually, from the context data extracted from DB server, low-level data processing services can infer, even against the policy, what is not supposed to be revealed at that stage, including user-ID. But, currently, we assume all the low-level data processing services are in ignorance of the user-ID, for the simplicity. In Fig. 2, the semi-transparent part between the DB server and the application service depicts the modules that do not know the user-ID, but handle the pseudo-user-ID. Thus leak of the user-id is, if exists, mainly responsible for policy description and management in DB server. Next section will introduce a policy description language for DB server.

Second, each step of information flows strictly follows predefined policies. For this, every module should be analyzed to get the set of the context data they use and yield [1]. On delivery of a context data, the policy manager checks if the flow is allowable. Especially, it will statically show possibility of context data leaks to composing the analysis results of the intermediate components in the chain for an information flow. The idea is not new [2] in other than the context data management. Next section contains the description of requirement description languages for software modules.

Third, maximize the degree of freedom of access context-data, as long as the privacy policies allows. Although this principle sounds natural, to follow it is not straightforward, since it is difficult to make a decision between the trade-off. In many cases, data really required by a service cannot be provided due to the privacy reasons. Or some users cannot help allowing suspicious services to access private data since they really want to have the service results.

As a good solution for the problem, the policy description language shown in the next section provides abstraction mechanism context data. That is, with the facilities to define policy on multiple views of context-data, services can continue working without leak of concrete private information.

3 Abstraction Description Model

3.1 A Context Data Model

Our policy languages are based on a context data model called UDM (Universal Data Model) elaborated by ETRI [1], but can be used for any other data models. UDM is similar to OEM [3], a semi-structured database model in that context information is denoted as an edge-labeled graph. The node represents an entity such as person, place, task, service, etc. Every node has its unique ID and type. There are special nodes that belong to “valued” type (grey circle in Fig. 3). These nodes can acquire values directly from Java objects. An association between nodes has direction, defined by which node is the from-node (Fnode) and which node is the to-node (Tnode).

Access to data in the context database is based path expression starting from given entry nodes. PLUE provides three entry nodes: $place, $task, and $owner. $task points to the root node from which application-specific data are described. $owner refers to the node from which the owner's information is represented, and $place implies the entry node of the place where the owner is placed. The $place points to the corresponding entry node of the current place of the user, who may move around different locations.

Using the path expressions starting from the designated entry nodes, a service is able to reach all the necessary context information, such as the title of the person who executes this application, the address of the place where the owner is currently placed, and other application-specific data. Fig. 2 shows an example of the UDM. $place.temperature and $place.air conditioner denoted for the thermometer and for the air conditioner in the place respectively where the user carrying out this application is located. Service invocation on a path is also possible, for instance $place.airconditioner.turnOn().

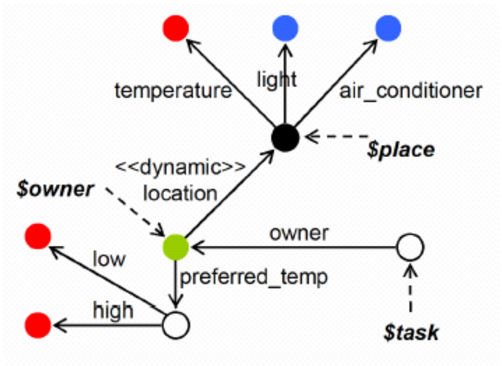


Fig. 2. An example of context data model in UDM [1]

3.2 Context-Data Abstraction Model

As mentioned in the previous section, our policy language provides data abstraction facilities in the policy description to provide services with maximal views of data without violating privacy. We consider an abstract data a set of selected nodes on a range of composite networks in UDM. With this approach, policy will not say "*Subject$_1$ is allowed to access object$_1$*", but say "*Subject$_1$ is allowed to access object$_1$ with an abstract-view$_1$*". Thus all subjects see the context-data through a kind of view.

To specify such abstraction, we propose a context-data abstraction model, for short CDAM. It consists of different kinds of description mechanisms, depending on the property of the data set. Let us start with a basic definition to describe the concept.

Def 1. *A 'composite object' is a meaningful connected sub-network of the entire data description in UDM consists of composite object.*

In Fig. 5, the black node representing $place and the edges to the two blue nodes and a red node for air-conditioner, light, and temperature respectively, can altogether make a composite node. One who defines the context-data model also defines composite objects. Such a composite object is a unit to be abstracted in CDAM. Two composite objects can share common sub-networks. Composite object can be nested. For simplicity, composite objects are generated as the data model is described, although they can be constructed dynamically at anytime, independent of the data themselves.

Def 2. *Each composite object has a representative node called 'central node', from which every node is reachable.*

If there exists no such node, a new node is added to the sub-network in UDM, with links to each node. Thus a node in the sub-network is nominated through a path from the central node. For instance, the black node in Fig. 5 can be the central node for the sub-network of four nodes and three edges from it.

Tagging edges with marks makes abstract views of the corresponding composite node. Following definition says that function T is used to tag an edge in a composite node.

Def 3. *T is the function from each edge in the composite object to a set of tags.*

If the edge is tagged to be invisible in an abstract view, the edge cannot be used to trace. Sub-networks unreachable from the central node in the composite object are hidden from the user through the abstract view.

Def 4. *t-edge-set is the set of edges of the composite object including the central node, tagged with a superset of {t}.*

Def 5. *The abstract view of tag t is the subnetwork consisting of the central node and the edges from t-edge set and reachable from the central node.*

Some edges with the abstraction tag cannot be included in the abstract view, since it is disconnected from the central node.

Def 6. *The entire data with a composite object as an abstract view of tag t is;*
- *The composite objects is replaced by the abstract view of tag t*
- *The edges outside of the composite object linked to its hidden edges are replaced by links to the nearest edges with tag t (in default) or the central node (when the nearest edges with t are farther than the central node).*

For the edges outside of the composite object, from/to the hidden edges are viewed from/to the nearest edges (in default) or the central node in this view, like a map with a reduced scale (The default can be changed by metadata description). The boundaries of composite objects ought to be carefully chosen.

The simplest case is that the data have only two abstract views-showing all or nothing. This case is the same as the one without abstraction. In this case, we define T such that 'for each edge e in a composite object c, T(e)={ACCESS}'. Then if a service that gets tag ACCESS for the composite object, it will be allowed to access the whole composite object; otherwise it cannot see anything about the composite object.

An abstract view can be defined by tagging some selected edges of the composite object. Let us assume that a person has her name, phone number, addresses and affiliation, and she wants to provide some service with her name and phone number. Then a tag PHONESERVICE will be added to the corresponding edges. T(e) will be {ACCESS, PHONESERVICE} for the name and the phone edges, but {ACCESS} for other edges. This means that there exist three kinds of services-allowed to access all about the context object, or only name and phone number data, or being prohibited from access any part of data. In this example, the tag ACCESS is not used in this example, but usually exists for the future services allowed with a transparent view.

For the example above, the tag TEMPERATURE can be used as follows, for automatic temperature control services.

- `T(light) = {ACCESS}`
- `T(airconditioner)={ACCESS, TEMPERATURE}`
- `T(temparature)={ACCESS, TEMPRERATURE}`

Then, only the services have TEMPERATURE tag in the policy can trace airconditioner and temparature edges. The tag ACCESS remains for the transparent view, again.

To model different kinds of actions like read and write, two kinds of tags are used such as READ and WRITE. For some e1's, T(e1) is equal to{ACCESS, READ} while for some other e2's, T(e2) is {ACCESS, WRITE}, when e1 is only readable and e2 is writtable. For the data e3 both readable and writable, T(e3) is {ACCESS, READ, WRITE}. For any private edge e4, T(e4) is defined as an empty set.

4 Policy Description Languages

In order to support high-quality services, context-aware systems deals with various real-life data, required to be dynamic and flexible. This makes the abstraction model for privacy defined on element-by-element base. However, when it comes to efficiency, some reduction is needed. Our policy language for privacy provides two directions of minimized description for simplicity. Each of them will be described in this subsection.

4.1 Schematic Description

The first one is schematic description. Although context data, by nature, do not have static schema, defining the 'kinds of context-data' precedes usual application development. It is not as strict as conventional database schema. Here we assume it is like a semi-structured-database schema entangled with mutual relationships. In fact, our UDM is similar to OEM, a semi-structured database model [3].

In a simple case, it can be represented XML schema, which is one of the common notations for context-data and semantic web [4]. Following example shows a location data, from the draft of Geographic Location/Privacy Working Group of the IETF [5].

```
<Location>
   <country> DE </country>
   <A1> Bavaria </A1>
   <A3> Munich </A3>
   <A4> Perlach </A4>
   <A6> Otto-Hahn-Ring </A6>
   <HNO> 6 </HNO>
</Location>
```

In UDM, the data is transformed as follows; the central node is generated with the XML tag <Location>. Other items in <Location> are directly linked with the central node.

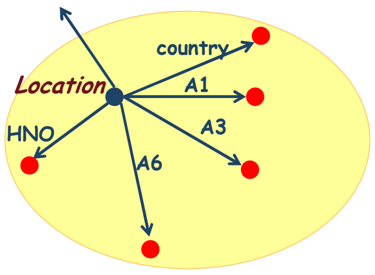


Fig. 3. Location data in UDM

For above data, it is natural to consider the abstract views from the hierarchical address format. With tags, the views are described as follows. The outermost XML tag transformed to be the central object is defined with a special attribute abstract with the value "YES", which means that it will work as a composite node. Each abstraction tag has a list of XML tags in a subitem named '<Data>'. Actually in UDM the element of the list represents links allowed for access. For instance, country, A1 A3 A4 can be seen a subject who is permitted to access the composite object as FOURTH.

```
<Composite name ="Location" abstract="YES">
    <Abstract tag = "SIXTH"> <Data link = "country"/>
      <Data link="A1"/><Data link="A3"/> <Data link="A4"/><Data link="A6"/>
      <Data link="HNO"/></Abstract>
    <Abstract tag = "FIFTH"> <Data link = "country"/>
      <Data link="A1"/><Data link="A3"/> <Data link="A4"/><Data link="A6"/>
      </Abstract>
    <Abstract tag = "FOURTH"> <Data link = "country"/>
      <Data link="A1"><Data link="A3"> <Data link="A4"/></Abstract>
    <Abstract tag = "THIRD"> <Data link = "country"/>
      <Data link="A1"/><Data link="A3"/> </Abstract>
    <Abstract tag = "SECOND"> <Data link = "country"/> <Data link="A1"/>
      </Abstract>
    <Abstract tag = "FIRST"> <Data link = "country"/> </Abstract>
</Composite>
```

To reduce the amount of policy description, we provide explicit definition of relationships between tags. Note that the tedious enumeration of tags in the above example is caused by the hierarchical relationships between the views. In this case, instead of enumeration, the policy can keep the linear ordering in <country>, <A1>,<A3>.<A4>, <A6> and <HNO>.

```
<Location abstract="YES">
<Abstract tag = "SIXTH" including="FIFTH"> <Data link="HNO"/> </Abstract>
<Abstract tag = "FIFTH" including="FOURTH"> <Data link="A6"/> </Abstract>
<Abstract tag = "FOURTH" including="FOURTH"> <Data link="A4"/> </Abstract>
<Abstract tag = "THIRD "including="THIRD"> <Data link="A3"/> </Abstract>
<Abstract tag = "SECOND" including="SECOND"> <Data link="A1"/> </Abstract>
<Abstract tag = "FIRST" > <Data link="country"/> </Abstract>
</Location>
```

Except FIRST, all the tag definitions are defined with "including" attributes, with some analogy with super class description of class definition in object-oriented languages. Compared to the tag definitions in the previous subsection, this succinct description has the relationships among abstraction tags explicitly. This disposes of the link-by-link (or XML tag-by XML tag) matching process that might be introduced for inclusion checking if the module analysis result does not break the policy in the DB server.

4.2 Policy Description

The actual policy is defined based on the above setup of abstraction. First, the DB server has a set of policy to control access to its data. For example, it can control data access by low data processing modules based on the following policy entries;

- `for all user.<module A, user.location,` *`SECOND`*`>`
- `for all user.<module B, user.location,` *`FOURTH`*`>`
- `for all user.<module C, user.location,` *`SECOND`*`>`
- `<module C,` $\texttt{user}_1$`.location,` *`FIRST`*`>`

The first tree of the policy entries have for all quantification, which can be omitted. The last entry appears with a data-owner 'user_1', for the sake of her privacy. Such a rule for a specific user can surpass those for general users.

As mentioned above, the analysis result on each low data processing module provides information about its behaviors related to data access. Following example shows the analysis result for module A, module B and module C, in a policy entry-like format.

- *`Module A :`* `<user.location, SECOND>`
- *`Module B:`* `<user.location, THIRD>`
- *`Module C:`* `<user.location, SECOND>`

Each module sees the user's location with an abstract view formed with abstraction tags including FIRST, SECOND, THIRD and etc. They are very similar to the maps with reduced scales.

For example, bound with SECOND, module A will access country information and A1 data. Since the DB server allows the module A to access location data with SECOND tag, it will access the required data without blocking. The module B will success too, since the accessible data from the analysis result (tagged with THIRD) are included in the data attached with FOURTH that are permitted to access module B. However, if analysis result of the module B is FIFTH, it will be blocked to access the data followed along 'A6' link (that is the data item in the XML tag 'A6'). The module C will fail when the user is user_1, but will success otherwise. Verification on such failure before execution will be helpful for the whole task to go well.

5 Related Works and Discussion

There have been various researches on privacy preserving mechanisms including anonymity, mixing, and cloaking for various software like routers and database systems [6]. Among them cloaking has similar approach to abstraction mechanisms, but rare of them generalize the idea to context data. Semantic e-Wallet [7] extends them for semantic web data, by allowing users to define separate 'obfuscation-by-abstraction' rules to hide details of context. However, not as a part of the access control policies, those rules are applied only to the result data extracted by queries. Thus the obfuscation-by-abstraction rules are either simple to be applied to result of various queries, or not be applicable to all query results. On the contrary, our abstraction mechanism supports expressive abstract views of the whole context data, embedded in access control policies.

Tags on data describing relevance levels are not new. In MLS (multi-level security) model [8], tags are used for clearance, by comparing the tags on subjects that request data access. However, they considered only single object, not a network of data, to be protected. Some XML security model [9] assigns to each node allowed operations such as 'R' and 'W', which may also be seen as one of the simplest cases of our abstraction tags.

Result of research on view management on semantic webs [10] may be helpful to abstract context data. A view description, usually based on a 'select-from-where' clause, seems to be corresponding to an abstraction category with a same tag in our approach. The difference is that a view describes data abstraction by condition indirectly, while our abstraction tags are attached to each edge of the data graph. We are planning to concentrate on combining those two kinds of description into our policy language for user convenience.

6 Conclusions

Customized services of context aware systems may invade users' privacy. To carry on services with privacy preserved, providing abstraction on the context data can be a good alternative.

This paper suggests an abstraction mechanism by tagging to data. A tag is corresponding to an abstract view of the context data. The abstraction behaves like a 'map' of the context data, with which privacy policy is described. Users, by assigning tags to data items, can express intended restriction on access from specific services.

We are planning to extend the policy languages to be more expressive, for instance, with constraints and indirect view description.

References

1. Eun-Sun Cho, Kang-Woo Lee, Security Checks in Programming Languages for Ubiquitous Environments, Workshop on Pervasive Security, Privacy and Trust, Aug. 2004
2. Andrew C. Myers., JFlow: Practical mostly-static information flow control. In *Symposium on Principles of Programming Languages (POPL)*, pages 228–241, Jan. 1999.
3. J. McHugh et al,. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3), September 1997.
4. K. Henricksen, J. Indulska, A. Rakotonirainy. Modeling context information in pervasive computing systems. In *1st International Conference on Pervasive Computing*, 2002.
5. Geographic Location/Privacy, http://www.ietf.org/html.charters/geopriv-charter.html

6. Gruteser, M., Grunwald, D. Anonymous usage of Location-Based Services Through Spatial and Temporal Cloaking. In 1st International Conference on Mobile Systems, Applications, and Services. May, 2003
7. F. L. Gandon, N. M. Sadeh, A Semantic e-Wallet to Reconcile Privacy and Context Awareness. In 2nd International Semantic Web Conference (ISWC), 2003.
8. Stallings, Cryptography and network security, Prentice Hall, 1999
9. Elisa Bertino and Elena Ferrari, Secure and selective dissemination of xml documents. *ACM Trans. Inf. Syst. Secur.*, 5(3):290–331, 2002.
10. Raphael Volz et al, Toward views in the Semantic Web. In Proceedings of the 2nd International Workshop on Databases, Documents, and Information Fusion (DBFUSION 02), July 4-5, 2002.

A Probabilistic Heuristic for Conflict Detection in Policy Based Management of Diffserv Networks

Hassine Moungla[1] and Francine Krief[2]

[1] LIPN Laboratory – UMR 7030 CNRS, University of Paris XIII, 99, Avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
Phone: +33 (0)1 49 40 28 41, Fax: +33 (0)148 26 07 12
hassine.moungla@lipn.univ-paris13.fr
[2] LaBRI Laboratory – UMR 5800 CNRS, University of Bordeaux I, 351, cours de la Libération, 33400 Talence, France
Phone: +33 (0)5 40 00 60 48, Fax: +33 (0)5 40 00 66 69
francine.krief@labri.fr

Abstract. Using Policy based management, the administration process can be simplified and largely automated. However, the conflict detection in specifying rules for policy-based networks remains an important problem to solve because the introduction of a new rule may contradict or duplicate each other. We propose a probabilistic heuristic method to guide the detection conflict algorithm. We present the performance results and an interactive java-based policy management tool editor which was designed for specifying the rules and detecting the conflicts among rules.

1 Introduction

Policy-based management has become a promising solution for managing enterprise-wide networks and distributed systems. These are typically large-scale systems, which require management solutions that are self-adapting and that dynamically change the behaviour of the managed system. The main motivation for the recent interest in policy-based services, networks and security systems is to support dynamic adaptability of behaviour by changing policy without recoding or stopping the system. This implies that it should be possible to dynamically update the policy rules interpreted by distributed entities in order to modify their behaviour. Furthermore, policies are rules governing the choices in behaviour of a system and can be used to define adaptable management actions. These policies thus define the conditions for performing a wide range of management actions such as a security or Quality of Service level change.

Policies are derived from business goals, service level agreements or trust relationships within or between enterprises. The refinement of these abstract policies into policies relating to specific services and then into policies implementable by specific devices supporting the service is not easy, and difficult to automate. Although the technologies for building management systems and implementing QoS and security are available, work on the specification and deployment of policies is still scarce. The precise and explicit specification of implementable policies is important in order to achieve the organisational goals using currently available technologies. The management of large sets of organisational objects (devices) across diverse boundaries often leads to conflicting requirements, which subsequently materialise into policy specification conflicts. Clearly, there is a limitation in the deployment of policy-based management models that do not provide a support for conflict detection and resolution. Some work on detecting conflict has also been initiated in the area of telecommunications feature interaction [1][2]. Inconsistency management in requirements analysis has also been investigated [3][4]; this work examines how useful the description of all system behaviour by means of a causal vocabulary would be in detecting conflict. Dulnop and all. discuss the critical nature of systems providing both a dynamic and static conflict detection and resolution and introduce a scalable computationally-efficient dynamic conflict detection mechanism[5]. These conflicts must be detected at each modification of the policy repository. Being able to detect conflict both statically and dynamically relies on a knowledge of some characteristics of the policy such as a policy conflict will only occur when the objectives of two or more active policies cannot be simultaneously satisfy[5]. Concerning the conflicts detection, it is important to specify an infrastructure of conflits detection and resolution. First this paper presents an overview of our policy model, and focuses on the policy definition and specification, in section 2. Section 3 proposes a probabilistic heuristic method which is used to guide our detection conflict algorithm. This algorithm is presented in section 4. Section 5 describes our conflicts resolution method. Section 5 shows the design of an interactive java-based policy rule editor with a module of rule conflict detection. Finally we conclude this paper and present the further work.

2 Policy Representation

This section provides the background to our current work on policy specifications and begins with an overview of our policy model. In the literature, we can find different opinions concerning the number of levels in a policy specification. This is sometimes called a policy hierarchy [6][7], and represents different views on policies, relationships between policies at different levels of this hierarchy, or abstractions of policies for the purpose of refining high-level management goals into low-level policy rules whose enforcement can be fully automated. The number of levels can be arbitrary but we accept three levels of policy Specification : *High-level/abstract* policies, *Specification-level* policies, *Low-level* policies or configurations.

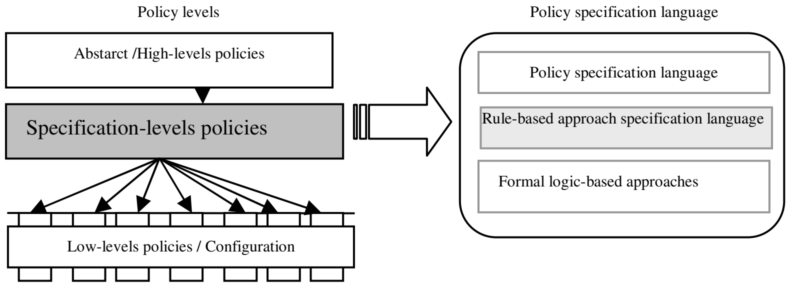

Policy levels
Abstarct /High-levels policies
Specification-levels policies
Low-levels policies / Configuration
Policy specification language
Policy specification language
Rule-based approach specification language
Formal logic-based approaches

Fig. 1. The approaches used for policy levels specification

Figure 1 summarises the approaches used for policy levels specification. To specify a policy, we can identify three main language categories: policy specification languages, rule-based specifications, and formal logic languages. The first approach concerns the specification of policies in a special language that can be processed and interpreted by a computer [8][9]. This approach maps a policy to a piece of software that can be executed by a computer under certain conditions [10] [11]. In the rule-based approach policies are specified as rules sequences of the form: *if condition then action*, and are mostly applied to the quality of service management. This approach is an intuitive method that can be easely used to traduce a Service Level Agreement (SLA) (IETF, DMTF). Finally, logic-based approaches are driven by the need to analyse the policy specification, but generally are not directly map to an implementation and are not easily interpreted by humans. Formal logic is mostly used in the security policies specification [5]. From a human standpoint, the best way to specify a high-level policy would be in terms of a natural-language in input. Although these policies are very easy to specify, the current state of natural-language processing, a special area within the field of artificial intelligence, needs to be improved significantly before such policies can be expressed in this manner.

A simpler approach is to interpret the policy as a sequence of rules, in which each rule is in the form of a simple condition-action pair (in a “if-then-else” format). The rules are evaluated on specific triggers, such as the time or a new packet arrival within the network. If a rule's condition is true, the action is executed. The IETF [12] has chosen a rule-based policy representation in its specification. However, due to the need to store this representation in an LDAP directory or a database, this representation essentially follows the tabular specification. The tabular representation represents each policy in an entry in a table (data base). The table consists of multiple attributes. Some of these attributes constitute the condition part, and the others constitute the action part. Different types of tables need to be specified if the condition components or action components of different rules vary. Furthermore, it is easier to analyze for the dominance and consistency.

There are several techniques and concepts to design and implement policies, but there is no commonly accepted terminology or notation. The proposal syntax is based on the IETF PCIMe and QPIM RFC. This syntax provides a model with a higher degree of flexibility for the definition of policy rules. However, this proposal allows for the level of flexibility and complexity that we want to give to the system, and eases the interoperation of the whole management system (e.g. through the definition of classes for variables and values).

This Information model follows the S*implePolicyAction* and *CompoundPolicyAction* structure proposed in PCIMe but it does not have the *PolicyVariableinPolicyAction* and *PolicyValueinPolicyAction* classes, since variables and values are simply expressed as properties of the specific *SimplePolicyAction* subclass.

The policies themselves will be expressed in an XML document that will contain the necessary classes that describe the policy rule. This XML document will be carried by our Policy-Manager-Tool interface, which will carry other important information, e.g. meta-policy rule and priority definition for the resolution module.

Why Using XML ?

Using XML as language for expressing policies has several advantages [13]. XML is ideal for transferring information between heterogeneous platforms because XML parsers are available for many platforms. Another advantage is that XML policy documents can be validated using an XML policy schema. This is possible because XML documents can carry a reference to their XML schema, instead of the schema itself. Moreover, the policy syntax checking functionality, is done intrinsically by the XML parser through the validation of the XML policy using its XML schema.

The mapping approach followed is based on a XML-Schema specification. All classes (e.g. PolicyConditionVariable, PolicyActionValue, SimplePolicyCondition, SimplePolice Action, etc. subclasses), properties, and even possible values, that are in conformity with our Information Model, in all domains, would be expressed in one or several XML Schema classes. With this approach, we are able to specify and check the correct policy syntax of any possible PolicyRule. We are even able to check that given property values are within some specific, valid range. As more details are given to the XML schema, more syntactic checks can be done to the policy by the XML parser.

Example1-XML rule representation **:**
Rule-name : Rule QoS_guarantee

IF the source address matches 192.168.1.0/24 **THEN** guarantee a minimum bandwidth of 1 Mbits

```
<POLICYGROUP>
<POLICYRULE name="QoS_guarantee"><content>
IF the source address matches 192.168.1.0/24
THEN guarantee a minimum bandwidth of 1 Mbits
</content>
IF <POLICYCONDITION>
 <COMPOUNDPOLICYCONDITIONCNF>
   <SIMPLEPOLICYCONDITION>
     <POLICYVARIABLE>
       <POLICYSOURCEIPV4VARIABLE>
        the source address
      </POLICYSOURCEIPV4VARIABLE>
    </POLICYVARIABLE>
 matches <POLICYVALUE>
    <POLICYIPV4ADDRVALUE>
       <IPADDRESS>
        192.168.1.0/24</IPADDRESS>
```

```
       </POLICYIPV4ADDRVALUE>
      </POLICYVALUE>
    </SIMPLEPOLICYCONDITION>
   </COMPOUNDPOLICYCONDITIONCNF>
  </POLICYCONDITION>
   THEN <POLICYACTION>
     <SIMPLEPOLICYACTION>
      <QOSPOLICYBANDWIDTHACTIONUNIT>
        guarantee a minimum bandwidth of
       <QPMINBANDWIDTH>  1 </QPMINBANDWIDTH>
       <UNIT>  Mbits</UNIT>
  </QOSPOLICYBANDWIDTHACTIONUNIT>
     </SIMPLEPOLICYACTION>
   </POLICYACTION>
  </POLICYRULE>
/POLICYGROUP>
```

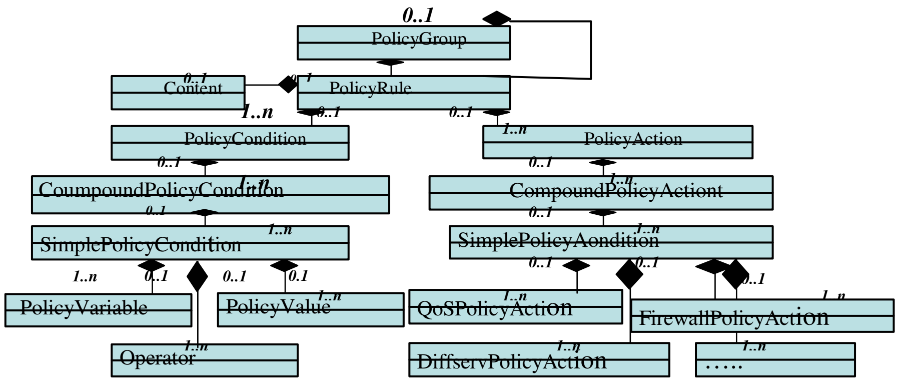


Fig. 2. A classes of our policy model

3 Probabilistic Heuristic of Conflicts Detection

A policy repository is a set of policy rules which are composed of two parts: a set of conditions and a set of actions. Conditions and actions can be seen as domain restrictions of an attribute, the restrictions number being usually small. Thus, a policy rule can be represented as a binary vector where each dimension indicates whether the corresponding attribute has been used or not.

Let Ci = { <attribute_ij, restriction_ij>; j in { 1, … ,K } } a set of conditions and A*i = { <attribute_ij, restriction_ij>; j in { 1, … ,L } }* a set of actions, where *restriction_j* is the restricted domain of *attribute_j*. Thus, $\Omega=\{Ri=<Ci, Ai>;\ i=1,\ldots,N\}$ is a set of *N* policies rules *Ri*.

Then, a rule *Ri* can be represented by a binary vector defined as follows:

$$\begin{cases} Ri = <ci1,ci2,\ldots,cik;\ ai1,ai2,\ \ldots,\ ail> \text{ with } \left[\begin{array}{l} \text{K condition variables and} \\ \text{L action variables} \end{array}\right. \\ \text{And} \\ cij = \begin{cases} 1 \text{ if } \textit{attribute\_j} \text{ appears in } Ci \\ 0 \text{ otherwise} \end{cases} \\ aij = \begin{cases} 1 \text{ if } \textit{attribute\_j} \text{ appears in } Ai \\ 0 \text{ otherwise} \end{cases} \end{cases}$$

Finally, this representation can be used to partition the rules description space according to their similarities. Therefore, a clustering algorithm, such as the topological maps [20,21], can be used to identify clusters (*Gi, i=1,…,t*) of similar rules description. This model starts with a small number of non-conflict rules. When a conflict occurs between two rules, the stored information is: the conflict rules, their clusters. First of all, the whole policy repository is tested using our algorithm detection (section4). Then, the probabilities guide the conflict detection algorithm. When a new rule *Rnew* is added, its cluster *Gi* is identified and the computed partition is used to evaluate the following conditional probabilities $P(conflict \mid R_{new} \in Gi \text{ and } R \in Gj)$ for each cluster in the policy repository without *Gi*.

$$P(conflict \mid R_{new} \in Gi \text{ and } R \in Gj) = \frac{\text{cardinality } \{(x, y) / x \in Gi,\ y \in Gj, \text{and conflict}\}}{\text{cardinality } Gi \times \text{cardinality } Gj}$$

A conflict probability is assigned to each pair of clusters. This probability is used to guide our conflict detection algorithm: the cluster having the highest probability is tested in priority.

4 Conflict Detection

Recent work [14][11][15] analysed the conflict types in distributed systems management and a study concerning the conflict detection in QoS management and security policies was presented in [11][16]. We focus on detecting rule conflict. Our results can be applied in this research area. When a new rule is defined, unintentionally conflicts may be introduced into the rules set. For example, if R1 and R2 rules are in the rules set: **Example2**.

R1 : **IF** the source address matches 192.168.1.1/24 **THEN** guarantee a minimum bandwidth of 1 Mbits

R2 : **IF** the source address matches 192.168.1.1/24 **and** the destination address matches 192.168.1.0/24 **THEN** guarantee a maximum bandwidth of 0.5 Mbits

The second rule will never be executed, because the actions parts are non-compatible.

A. Syntactic versus Semantic Conflicts (method)

It is useful to make a distinction between two broad classes of conflicts. Dourish [17] classifies conflicts as either *syntactic* or *semantic*. Syntactic conflicts represent the inconsistencies that occur below the level of application code; that is, in the toolkit and systems infrastructure itself. Semantic conflicts are the inconsistencies that occur above the dividing line between application and infrastructure. A conflict detection module should detect this type of errors and advertise the administrator with informative messages.

B.Conflict detection Algorithm

Policy conflict occurs when the objectives of two or more policies can not be simultaneously satisfy (see example2). We suppose that each condition part of rule represents a set. The R2 rule is in conflict with the first one. The condition part of R2 represents a subset of condition part of the R1 rule. To detect this kind of conflicts, we need to know the relationship between two condition parts of rules. There are four potential relations type, which interests us:

1. cond1 $\subset$ cond2 (cond2 contains cond1)
2. cond2 $\subset$ cond1 (cond1 contains cond2)
3. cond1 $\cap$ cond2 $\neq \emptyset$ (cond1 intersects with cond2)
4. cond1 $\cap$cond2 $= \emptyset$ (cond1 disjoins with cond2)

To detect the conflicts, there are two steps. First, we must know how to detect the “belong” relation. The 1 and 2 cases presented previously can be expressed with the relation “belong”. Then, we must know how to detect the “intersects” relation. The second step is to check the compatibility of the actions.

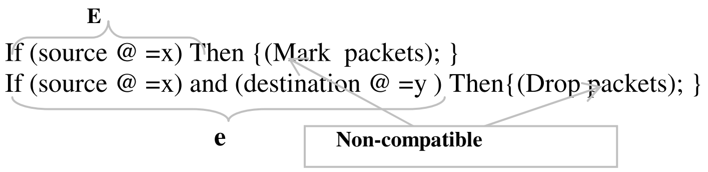
Definition 1. The condition part of the policy rules can be represented by a set which respects some characteristics and on which one or more actions are applied. The enforcement of another actions on a sub-set of this set can provoke a conflict (the actions parts must be checked). Tables (data-base) are created to specifies if cond1 $\subset$ cond2 (Table1) or cond1 $\cap$ cond2 (Table2).

Definition 2. Two rules are non-compatible if the two rules establish the act and its opposite. To check this compatibility, we create a data-base of compatibility actions, for example we have defined a table for non-compatibility actions in a Diffserv network environment.

Table 1. Table of the non-compatible actions in a Diffserv network

Actions	Shape	Mark	wMax	BwMin	Drops
Shape	Y	N	N	N	N
Mark	N	N(≠)	N	N	N
BwMax	N	Y	Y	N	N
BwMin	N	Y	N	Y	N
Drops	N	N	N	N	Y

N : Non-compatible **Y** : compatible



```
If (source @ =x) Then {(Mark  packets); }
If (source @ =x) and (destination @ =y ) Then{(Drop packets); }
```

Definition 3. To check if condition 1 intersects with condition 2, we associate a variables-condition table, which invokes this intersection condition-part. Example: POLICYSOURCEIPV4VARIABLE and an other POLICYSOURCEIPV4 VARIABLE with the same POLICYIPV4ADDRVALUE.

Algorithm

Let rule1: If **cond1** Then **act1,** rule2: If **cond2** Then **act2**

E a set which represents the Set-**cond1** of **rule1, e** a set which represents the Set-**cond2** of **rule2**

```
Begin- Conflict detection
  { Check the condition parts of rules with appropriate
       functions
  if [ E(cond1)  ⊃e(cond2) ] (with Table1- see definition1)  then {call Compatible function to be
     able to check if the actions are compatible (check the data-base compatibility)
              • If compatibility is ok there are no conflict
              • If compatibility is not ok there is a conflict, the algorithm sends a message to the Policy
                manager.
       }
 Else {
        If [ E(cond1) ∩  e(cond2) ] (with Table2- definition1)  then { call Compatible
     function to be able to check if the actions are compatibles
              • If compatibility is ok there are no conflict
              • If compatibility is not ok there is a conflict, the algorithm sends a message to the Policy
                manager. }
       }
 }
```

5 Conflict Resolution

After the identification of a conflict, the Policy Manager Tool receives a message with the rules which are in conflict. It is necessary to eliminate entirely the conflict. However, it is not always possible, so we identify three possible approaches of conflicts resolution: Total elimination; Prioritization; Dynamic elimination.

Elimination implies the modification of the conflicting statements so that the conflict cannot occur. This must not, of course, be done unilaterally but has to be renegotiated. For example, a new SLA can be renegotiated using a negotiation protocol such as SLN-NSLP[19].

Policy rules can be associated with a priority value to resolve conflicts between rules. This approach is not scalable in large networks with a large number of rules specified by different administrators. In addition, policy rules can be tagged with one or more roles. A role represents a functional characteristic or the capability of a resource to which policies are applied, such as a backbone interface, a frame relay interface, a BGP-capable router, a web-server, a firewall, etc. The use of role labels is essentially used as a mechanism for associating policies with the network elements to which the policies apply. If prioritization is not a solution to resolve the conflict, the Policy Manager can define meta-policies to resolve the conflict problem. Meta-policies are policies about how policies are interpreted and how conflicts are resolved. Conflicts only occur if the policies are about the same action, on the same target but the modalities (right/prohibition, obligation/dispensation) are different. For example, one policy could give the user with the X IPaddress the right to have the maximum bandwidth and the another policy could prohibit the same user with the same IPaddress from to have this bandwidth. Meta policies in our system regulate conflicting policies in two ways; by specifying priorities and by using precedence relations [18].

6 Implementation

We have implemented an interactive Java-based Policy Management Tool editor, for editing the QoS policies rules set and detecting the conflicts between the rule being specified and the existing rules set. If a conflict is detected, a message is transmitted directly to the Management Tool in order to resolve it. The Policy Manager can enter the rules using a semi-natural language. Then, these rules are translated into an XML format and controlled by a simple menu which provides not only the various fields constituting a policy but also integrates techniques for the syntactic checking using a XML parser . The rules are stored in the policy repository.

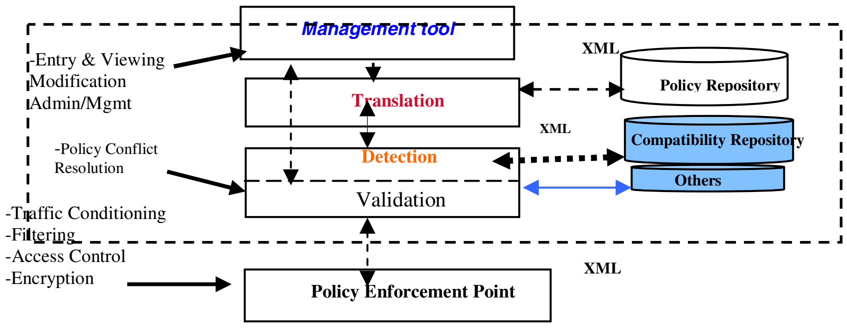
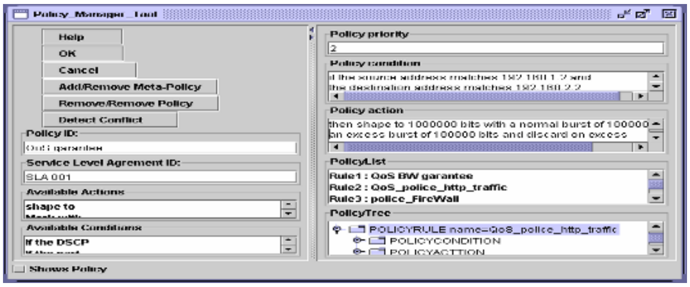


Fig. 3. Architecture of a Policy-Based Management Tool



Policy ID:
QoS guarantee

Service Level Agrement ID:
SLA 001

Available Actions
shape to

Available Conditions
If the DSCP

Policy priority
2

Policy condition
if the source address matches 192.168.1.2 and
the destination address matches 192.168.2.2

Policy action
then shape to 1000000 bits with a normal burst of 100000
an excess burst of 100000 bits and discard on excess

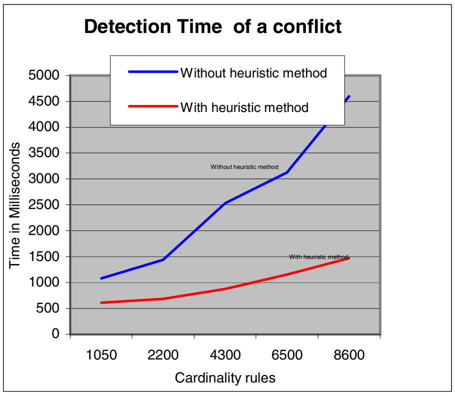
PolicyList
Rule1 : QoS BW garantee
Rule2 : QoS_police_http_traffic
Rule3 : police_FireWall

PolicyTree
POLICYRULE name=QoS_police_http_traffic
POLICYCONDITION
POLICYACTTION

Fig. 4. Java Management Tool Editor

Figure 4 shows the screen-dump of the Policy Management Tool editor. It can open the text file of an existing rule or a meta-policy rules set, and load the rules set into its internal XML structure, and check the coherence with the other rules.

Figure 5 and 6 shows the performance results. These results are calculated with five clusters: *G1, G2, G3, G4, G5*. The new rule which is tested belongs to *G1* cluster. The probability that a conflict occurs with each cluster type is calculated, the highest probability is between *G1 and G2*. Two test types are computed : *the detection time of a conflict* and *the number of rules tested.* Each test type is computed with and without our probabilistic heuristic. We can see that the probabilistic heuristic improves the performances (Figures 5 and 6). Below 1000 rules, the user do not see a noticeable delay. The tests results are based on the existence of one conflicting policy rule in the policy repository.



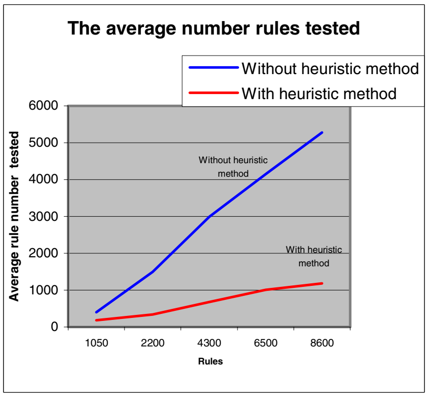


Fig. 5. Performance of our conflict detection Method

Fig. 6. The rules number tested with our. conflict detection method

7 Conclusion and Future Work

A flexible policy-based management framework, an conflicts detection algorithm and a conflicts resolution method have been presented in this paper. The policies class considered is limited to Diffserv networks. We are currently extending this approach and performing experimental results by taken into account a probabilistic heuristic method and a new method of clustering. This work will be the subject of a forthcoming submission.

References

1. "Special Issue on Feature Interactions in Telecommunications Systems", IEEE Communications Magazine, Volume 31, Number 8, 1993.
2. M. Thomas, "Modelling and Analysing User Views of Telecommunications Services", Feature Interactions in Telecommunications Networks, IOS Press,1997.
3. J. D. Moffett, J. G. Hall, A. C. Coombes, J. A. McDermid, "A Model for a Causal Logic for Requirements Engineering", Volume 1, Number 1, Journal of Requirements Engineering, 1996.
4. J. D. Moffett, A. J. Vickers, "Behavioural Conflicts in a Causal Specification", Volume 7, Number 3, Automated Software Engineering, 2000.
5. N. Dunlop, J. Indulska, K. Raymond, *"Dynamic Conflict Detection in Policy-Based Management Systems"*, IEEE Enterprise Distributed Object Computing Conference (EDOC'2002), Lausanne, Sept 2002.
6. Moffett, J. D. and M. S. Sloman (1993). *Policy Hierarchies for Distributed Systems Management*. IEEE JSAC Special Issue on Network Management, vol. 11(9), pp. 1404-1414, December 1993.
7. Weis, R. (1994a). *Policy Definition and Classification: Aspects, Criteria and Examples*. In Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations & Management, Toulouse, France, 10-12 October 1994.
8. J. Fritz Barnes and Raju Pandey. ``CacheL: Language Support for Customizable Caching Policies. Proceedings of the Fourth International Web Caching Workshop (WCW '99), March 1999, San Diego, California, USA.
9. James Hoagland, "Specifying and Implementing Security Policies Using LaSCO, the Language for Security Constraints on Objects". Ph.D. Dissertation, University of California, Davis, March 2000.
10. Dinesh C. Verma, *Member, IEEE* Simplifying Network Administration using Policy based Management
11. Nicodemos C. Damianou, "A Policy Framework for Management of Distributed Systems", Ph.D. Dissertation. Imperial College of Science, Technology and Medicine University of London, Department of Computing-London, February 2002.
12. http://www.ietf.org/html.charters/policy-charter.html.
13. M. Brunner, "Service Management in a Telecom Environment based on Active Network Technology", ETH Zurich, TIK-Schriftenreihe No. 34, Diss. ETH Zurich No. 13433, November, 1999.
14. Chomicki, J., J. Lobo and S. Naqvi (2000). *A Logic Programming Approach to Conflict Resolution in Policy Management.* In Proceedings of the 7th International Conference in Principles of Knowledge Representation and Reasoning, Breckenridge, Colorado, USA, Morgan Kaufmann Publishers, pp. 121-132, April 2000.
15. E. C. Lupu, M. S. Sloman, "Conflicts in Policy-Based Distributed Systems Management", IEEE Transactions on Software Engineering - Special Issue on Inconsistency Management, 1999

16. Florin Baboescu, George Varghese,“Fast and Scalable Conflict Detection for Packet Classifiers”, Proceedings of the 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT’02) IEEE.
17. Dourish, P. *Open Implementation and Flexibility in Collaboration Toolkits.* Ph.D. dissertation, Computer Science Department, University College, London, June, 1996.
18. Emil C. Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering*, 25(6):852–869, November/ December 1999.
19. N. Mbarek anf F. Krief, "SLN NSLP: A Service Level Negotiation Signaling Protocol in the NSIS Environment", ICT'2005.
20. Diday, E. C,Simon: Clustering Analysis, in :K.S. Fu(ED), Digital pattern recognition, Springer,New York.An Introduction to Symbolic Data. (1996).
21. Lebbah, M. Badran, F. Thiria: S Topological Map for Binary Data , ESANN 2000 , Bruges, April 2000.

LEA2C: Low Energy Adaptive Connectionist Clustering for Wireless Sensor Networks

Lahcène Dehni[1], Younès Bennani[1], and Francine Krief[2]

[1] Laboratoire d'Informatique de Paris Nord,
UMR 7030 du CNRS, Université Paris 13,
99, avenue J.B. Clément - 93430 Villetaneuse
{ldehni, younes}@lipn.univ-paris13.fr
[2] LAboratoire Bordelais de Recherche en Informatique,
UMR 5800 du CNRS, Université Bordeaux 1,
351, cours de la Libération - 33405 Talence Cedex
krief@labri.fr

Abstract. The use of the wireless sensor networks (WSNs) should be increasing in different fields. However, the sensor size is an important limitation in term of energetic autonomy, and thus of lifetime because battery must be very small. This is the reason why, today, research mainly carries on the energy management in the WSNs, taking into account communications, essentially. In this context, we propose an adaptive routing algorithm based on the clustering that we named LEA2C. This algorithm relies on connectionist learning techniques and more exactly on the topological self-organizing maps (SOMs). New rules for the choice of the clusterheads have also been added. By comparing the results obtained by our protocol with those of other clustering methods used in the WSNs, such as LEACH and LEACH-C, we obtain important gains in term of energy and thus of network lifetime.

1 Introduction

WSNs present a vast application field: scientific, logistic, military and health field... According to MIT'S Technology Review, this technology is one of the ten new technologies which will change the world and our manner of live and work [1].
The battery is an important component of a sensor. Generally, it is neither replaceable nor rechargeable. With its small size, it provides a very limited energy quantity [2]. So, it limits the sensor lifetime and influences the total network operation. This is the reason why, today, protocols ensuring low energy consumption occupy an important research orientation in this field.

A sensor ensures acquisition, data processing and communications. These last are the most energy consuming. Thus, a good diagram of energy management must, in priority, take into account communications. The majority of communication protocols in the Ad-Hoc networks do not satisfy the characteristics of the WSNs. This is the reason why there is a need for improving them or developing new ones. The two principal classes of protocols used in the wireless networks are based on the multi-hops routing or clustering techniques (or a hybridization of two techniques). Several approaches are proposed to calculate the optimal path in the multi-hops routing protocols. Some propose to take into account the shortest path, in term of distance, to the base station [3]. Others techniques privilege the sensors having the maximum energy levels [4]. However, the main disadvantage of the multi-hops routing is the periodicity of the messages sent to maintain the valid paths. These messages overload the network and consume additional energy.

Today, clustering gives the best results [5]; this is the reason why we adopted this approach. It is a problem of classification which interests the numerical learning for a long time, in particular, the connectionist models and, more particularly, the topological Self-Organizing Maps (SOMs). This last approach has proved its efficiency in this type of problem. In this paper, we propose to adapt an unsupervised connectionist learning method by introducing the evolutionary and dynamic clustering aspect. This new approach of evolutionary clustering allows us improving the efficiency of the routing in WSNs. The proposed routing algorithm gives better results than those obtained by the LEACH (Low-Energy Adaptive Clustering Hierarchy) [6] and LEACH-C (Low-Energy Adaptive Clustering Hierarchy - Centralized) [7] protocols, which use other methods of clustering.

Our paper is organized as follows: first, we present an energy consumption model in the WSNs. Then, we study two routing protocols based on the clustering that we use to compare our results. We, also, present the unsupervised numerical learning technique for the clustering, called SOMs. The adaptation of this classification technique to the routing problem allowed us to propose a routing algorithm adapted to the WSNs. Finally, we show, through an experiments series, some validation results of our new algorithm and we present the future prospects.

2 Energy Consumption Model in WSNs

A sensor uses its energy to carry out three main actions: acquisition, communication and data processing.

The power consumption to perform the data acquisition is not very important. Nevertheless, it varies according to the phenomenon observed and monitoring type. The communications consume much more energy than the other tasks, in emission as well as in reception (Fig. 1.).

To transmit a *k* bits message over a distance of *d* meters, the transmitter consumes:

$$E_{Tx}(k,d)=E_{Tx}(l)+E_{Tx\_amp}(k,d) \tag{1}$$

$$E_{Tx}(k,d)=\begin{cases} k.E_{elec}(k,d)+k.\varepsilon_{friss}.d^2 & \text{if} \quad d<d_{crossover} \\ k.E_{elec}(k,d)+k.\varepsilon_{two\_ray\_amp}.d^4 & \text{else} \end{cases} \tag{2}$$

To receive a *k* bits message, the receiver consumes:

$$E_{Rx}(k)=E_{Rx\_elec}(k)=k.E_{elec} \tag{3}$$

Where : E_{elec} : energy of electronic transmission/reception; k : size of a message; d : distance between the transmitter and the receiver; E_{Tx_amp} : amplification energy; ε : amplification factor; $d_{crossover}$: limit distance over which the transmission factors change their value.

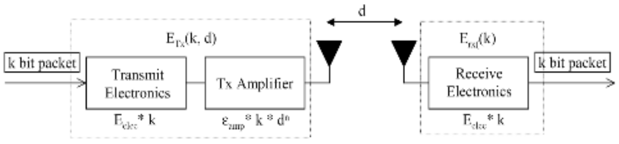


Fig. 1. Communication power consumption model in WSNs [15]

Power consumption for the calculation operations is much weaker than the communication energy. An example is cited in [2]. The necessary energy to transmit a message of 1 KB over a distance of 100m is roughly equivalent to the necessary energy consumed to carry out 3 million instructions with a speed of 100 million instructions per second (MIPS). This level can be exceeded according to the installed circuits in the sensors and the necessary functionalities. Thereafter, it is calculated by applying the following:

$$E_{DA} = 5nJ/Bit/msg \tag{4}$$

3 LEACH and LEACH-C Protocols

We retained the works of Heinzelman and al. [6,7] because they have excellent results in term of power economy. The authors have proposed the LEACH protocol and a centralized version of this protocol, called LEACH-C. These protocols are based on clustering (Fig. 2.).

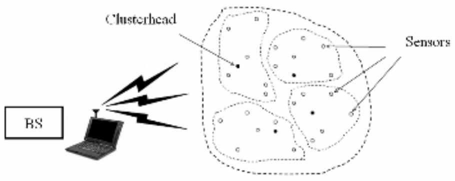


Fig. 2. Clustering in WSNs

Clustering consists in the segmentation of the network into groups (clusters). Sensors transmit their data towards group representatives called clusterheads (CHs), which send these data to the base station (BS). In some applications, CHs make a simple data processing (aggregation …) on the received data before retransmitting them to the BS. This approach permits the bandwidth re-utilization. It also offers a better resource allocation and helps to improve the power control in the network [8,9].

In LEACH protocol, data aggregations, compressions and routing minimize energy consumption by reducing the data flow and thus the total communications. The sensors are homogeneous and have the same energy constraints. Clustering allows sensors to establish small communication distances with their CHs.

The system is based on a probabilistic demand models and fixes the optimal number of clusters according to some parameters such as the network topology, the communications and the computational cost. Generally, CHs represent 5% of the sensors number in the network. They receive the answers from simple sensors and create TDMA tables according to the number of sensors in a cluster.

Each sensor transmits its data to its CH using the time slots specified in the TDMA tables. Sensors turn of their antennas and wait for their speaking time. This method permits to minimize the energy dissipation. CHs leave their receivers on to assure the reception of all the sensors data. Each CH chooses randomly a code in a list of CDMA propagation codes; it transmits this code to its cluster sensors. The sensors use this code for their transmissions. This method permits to minimize the communication interferences between closed CHs. The clusters are self-configured independently of the BS (distributed algorithm). With the TDMA/CDMA techniques, a hierarchy built on a multilevel clustering can be built and used to increase the saved energy amount.

LEACH presents the following disadvantages:

Optimal rate of CHs is set to 5% without any justification; nevertheless, networks have their own topology, density and number of sensors, and might be different from each others. Any suggestion concerning the time of CHs reelection (iterations period) has been done and the further a CH is from the BS, the more quickly it dies.

An extension of this algorithm, called LEACH-C [7], has been proposed to avoid these drawbacks. In this centralized iterative algorithm, the clusters structure is computed by the BS using the "Simulated annealing" optimization method [10]. At each step, the network sensors are given their role, either CH or simple sensor, by the BS. Then, operations continue same as in the LEACH protocol.

4 The Unsupervised Connectionist Learning and the SOM

Unsupervised numerical learning, or automatic classification, consists in determining a partition of an instance space from a given set of observations, called training set. It aims to identify potential trend of data to be gathered into classes. This kind of learning approach, namely clustering, seeks for regularities from a sample set without being driven by the use of the discovered knowledge. Euclidian distance is usually used by clustering algorithms to measure similarities between observations.

SOMs implement a particular form of competitive artificial neural networks (ANN). When an observation is recognized, activation of an output cell – competition layer – leads to inhibit activation of other neurons and reinforce itself. It is said that it follows the so called “Winner Takes All” rule. Actually, neurons are specialized in the recognition of one kind of observations. The learning is unsupervised because neither the classes nor their number is fixed a priori.

A SOM consists in a two dimensional layer of neurons (Fig. 3.) which are connected to n inputs according to n exciting connections of respective weights w and to their neighbors with inhibiting links.

The training set is used to organize these maps under topological constraints of the input space. Thus, a mapping between the input space and the network space is constructed; two closed observations in the input space would activate two closed units of the SOM.

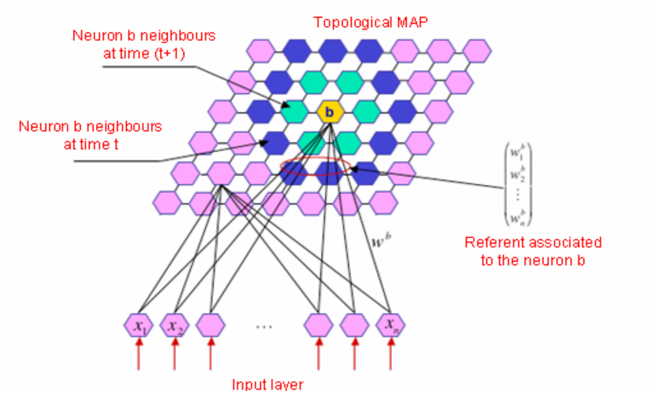

Topological MAP
Neuron b neighbours at time (t+1)
Neuron b neighbours at time t
b
w^b
Referent associated to the neuron b
x_1
x_2
...
x_n
Input layer

Fig. 3. MAP structure (2D network)

An optimal spatial organization is determined by the ANN from the received information, and when the dimension of the input space is lower than three, both position of weights vectors and direct neighborhood relations between cells can be represented visually. Thus, a visual inspection of the map provides qualitative information of the map and the choice of its architecture.

The connectionist learning is often presented as a minimization of a risk function. In our case, it will be carried out by the minimization of the distance between the input samples and the map prototypes (referents), weighted by a neighborhood function h_{ij}. To do that, we use a gradient algorithm. The criterion to be minimized is defined by:

$$E_{SOM} = \frac{1}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} h_{jNN(x^{(k)})} \left\| w_{.j} - x^{(k)} \right\|^2 \tag{5}$$

N represents the number of learning samples, M the number of neurons in the map, $NN(x^k)$ is the neuron having the closest referent to the input form $x^{(k)}$, and h the neighborhood function.

The neighborhood function h can be defined as:

$$h_{rs} = \frac{1}{\lambda(t)} \exp\left(- \frac{d_1^2(r,s)}{\lambda^2(t)} \right) \tag{6}$$

$\lambda(t)$ is the temperature function modeling the neighborhood extent, defined as:

$$\lambda(t) = \lambda_i \left(\frac{\lambda_f}{\lambda_i} \right)^{\frac{t}{t_{\max}}} \tag{7}$$

λ_i and λ_f are respectively initial and the final temperature (for example λ_i= 2, λ_f = 0.5). t_{max} is the maximum number allotted to the time (number of iterations for the x learning sample). $d_1(r,s)$ is the Manhattan distance defined between two neurons r and s on the map grid, with the coordinates (k,m) and (i,j) respectively:

$$d_1(r,s)=|i-k|+|j-m| \tag{8}$$

The learning algorithm of this model proceeds essentially in three phases:

- **Initialization phase** where random values are assigned to the connections weights (referents or prototypes) of each neuron of the map grid.
- **Competition phase** during which, for any input form $x^{(k)}$, a neuron $NN(x^k)$, with neighborhood $V_{NN(x^{(k)})}$, is selected like a winner. This neuron has the nearest weight vector by using Euclidean distance:

$$NN(x^{(k)})=\underset{1\le i\le M}{\operatorname{argmin}}\left\|w_{.i}-x^{(k)}\right\|^2 \tag{9}$$

- **Adaptation phase** where the weights of all the neurons are updated according to the following adaptation rules:

If $w_{.j}\in V_{NN(x^{(k)})}$ then adjust the weights using:

$$w_{.j}(\mathrm{t}+1)=w_{.j}(\mathrm{t})\text{-}\varepsilon(t)h_{jNN(x^{(k)})}\left(w_{.j}(\mathrm{t})\text{-}x^{(k)}\right) \tag{10}$$

else

$$w_{.j}(\mathrm{t}+1)=w_{.j}(\mathrm{t}) \tag{11}$$

Repeat this adjustment until the SOM stabilization.

SOM map segmentation:

We segment the SOM using the K-means method (Fig. 4.a.). It is another clustering method. It consists in choosing arbitrarily a partition. Then, the samples are treated one by one. If one of them becomes closer to the center of another class, it is moved into this new class. We calculate the centers of new classes and we reallocate the samples to the partitions. We repeat this procedure until having a stable partition.

The criterion to be minimized in this case is defined by:

$$E_{K-means}=\frac{1}{C}\sum_{k=1}^{C}\sum_{x\in Q_k}\left\|x-c_k\right\|^2 \tag{12}$$

Where C represents the number of clusters, Q_k is the cluster k, c_k is the center of the cluster Q_k or the referent.

The basic algorithm requires fixing K, the number of wished clusters. However, there is an algorithm to calculate the best value for K assuring an optimal clustering. It is based principally on the minimization of Davies-Bouldin index [11], defined as follows:

$$I_{DB}=\frac{1}{C}\sum_{k=1}^{C}\max_{l\neq k}\left\{\frac{S_c(Q_k)+S_c(Q_l)}{d_{ce}(Q_k,Q_l)}\right\} \tag{13}$$

With

$$S_c(Q_k)=\frac{\sum_i \left\|x_i-c_k\right\|^2}{|Q_k|} \tag{14}$$

and
$$d_{cl}(Q_k,Q_l)=\|c_k-c_l\|^2 \tag{15}$$

C is the number of clusters, S_c is the intra-cluster dispersion, and d_{cl} is the distance (centroid linkage) between the clusters centers k and l. This clustering procedure aims to find internally compact spherical clusters which are widely separated.

There are several methods to segment the SOMs [12]. Usually, they are based on the visual observations and the manual assignment of the map cells to the clusters. Several methods use the K-means algorithm with given ranges for K value. Our work is based on the approach of Davies-Bouldin index minimization.

We note that the K-means approach can be applied directly to the data instead of SOMs. In our work, we applied it to the SOMs results. The idea is to use SOMs as a preliminary phase in order to set a sort of data pretreatment (dimension reduction, regrouping, visualization...). This pretreatment has the advantage to reduce the clusters calculation complexity and also ensure a better visualization of the automatic classification results. Moreover, the use of SOMs for visualization is crucial, especially in the case of multivariate data: dimension > 2 or 3. In this last case, the SOMs permit, on one hand, to reduce the data space dimension, and in other hand, to visualize the clusters in the plan.

5 LEA2C: A New Routing Approach in WSNs

We use clustering methods based on the unsupervised connectionist learning techniques and different properties of LEACH-C in order to propose a new routing approach in WSNs, so called LEA2C (Low Energy Adaptive Connectionist Clustering).

Our approach is based on the principal following axes:

- The algorithm is iterative: by using the SOMs, the BS calculates the clustering according to coordinates of the alive sensors and assigns roles to the sensors in each iteration.
- Like in LEACH-C, TDMA and CDMA techniques are used to avoid the interferences and we use the same energy consumption model Fig. 1. [7].
- We propose that in each cluster, the CH choice can be made using one of the three following criteria:
    - CH is the sensor having the maximum energy level in the cluster.
    - CH is the nearest sensor to the gravity center of the cluster (physical distances).
    - CH is the nearest sensor to the BS.

LEA2C algorithm:

To compare our approach with LEACH and LEACH-C methods we use the same communication and data model.

1) Initialization

Random deployment of N=100 homogeneous sensors in a given space (100x100), with the same energy level (E = 2J). The BS coordinates are (50x300).

2) Clustering
2.1) By using the SOM algorithm, a first sensors clustering is calculated, based on the sensors coordinates. Also, the K-means clustering is applied on the SOMs results.
2.2) CHs are selected in all the clusters according to one of the selection criteria proposed above.
2.4) A role is allotted to each sensor (CH or simple sensor).

3) Data transmission
3.1) Simulation of the data transmission from the simple sensors to their CHs: consumed energy is calculated using the formulas (1) and (2).
3.2) We calculate, for each CH:
- The data reception energy using the formula (3)
- The data aggregation energy using the formula (4)
- The results transmission energy to the BS by using the formula (1).
3.3) Reelection of the CHs according to the chosen selection criterion. When the CHs are not chosen according to the first criterion (maximum energy), the same CHs are reelected after each transmission until having fewer power level in the cluster.
3.4) Repeat the steps 3.1) to 3.4) until the dead of a sensor because of the energy lack or a failure.

4) *Repeat* the steps 2) and 3) until the death of all the sensors in the network.

6 Simulations Results

To simulate our algorithms, we have used SOMs core (SomToolbox) proposed by HUT (Helsinki University of Technology) researchers [13]. We have applied our algorithm on the same data retained in [7]. The figures below present the results obtained.

Figure 4.a. represents the neurons segmentation of the SOM map.

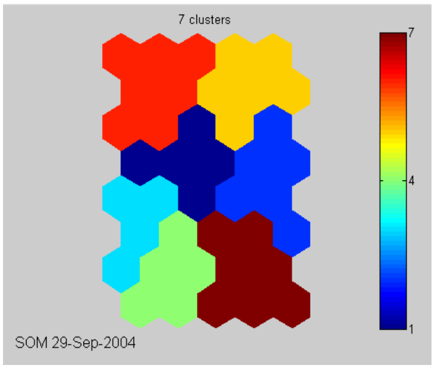


Fig. 4.a. SOM clustering

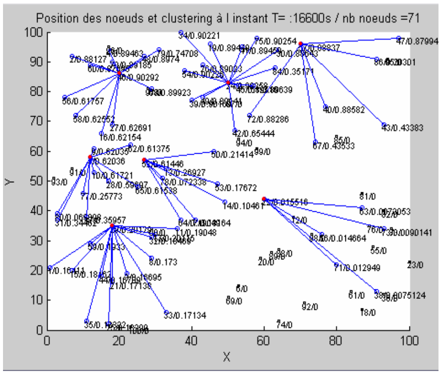


Fig. 4.b. LEA2C communications between nodes

Figure 4.b. represents the sensors network, and shows the links between the CHs and their sensors. Failing sensors are represented by black squares. The identifier and the energy level of each sensor are indicated at the given moment. Sensors positions on the figure reflect their real geographical positions.

Figures 5.a, 5.b and 5.c represent respectively the variation of the number of alive sensors according to the number of received messages by the BS, the variation of the number of alive sensors according to the time and the variation of the number of messages according to the time. In each figure, we consider the corresponding variations according to the four scenarios:

- Direct transmission to the BS: each sensor communicates its data directly.
- The clustering according to our approach with the three CH selection criteria already mentioned:
- The choice of nearest sensor to the gravity centre (GC) of the cluster.
- The choice of nearest sensor to the BS.
- The choice of the sensor having the maximum energy level in the cluster.

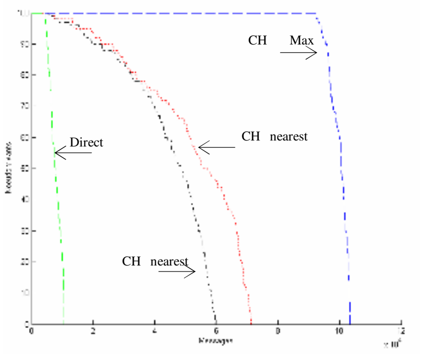


Fig. 5.a. Number of the alive motes VS number of messages received by the BS with LEA2C

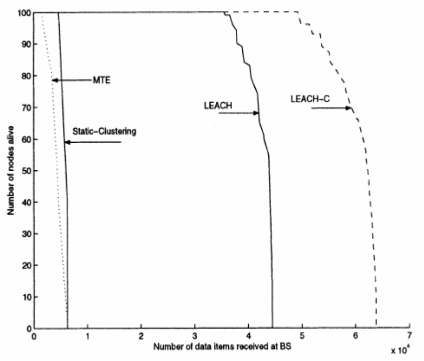


Fig. 6.a. Number of the alive motes VS number of messages received by the BS with LEACH and LEACH-C [7]

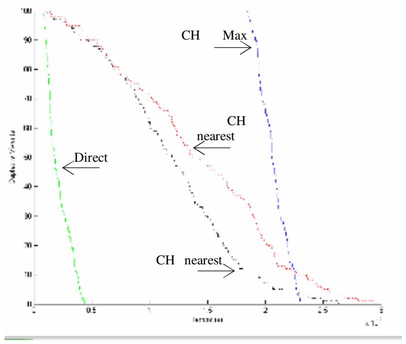


Fig. 5.b. Number of the alive motes VS time with LEA2C

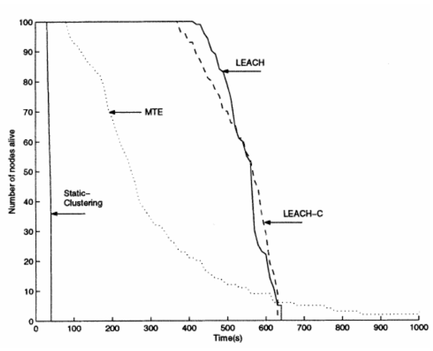


Fig. 6.b. Number of the alive motes VS time with LEACH and LEACH-C [7]

Figures 6.a, 6.b and 6.c represent the same variations for the following protocols: a MTE multi-hop routing protocol [14], a fix clustering and the LEACH and LEACH-C protocols.

In general, figures 5.a, 5.b and 5.c show, for our algorithm, that the choice of the CH according to the criterion of maximum energy is better than the choices according to the other criteria. Choices according to the criteria of proximity to the GC and to the BS have nearly the same performances. They are enough far from the criterion of maximum energy but better than the performances of the direct transmission protocol.

Figures 6.a, 6.b and 6.c show the effectiveness of the LEACH-C protocol compared to the standard LEACH protocol. Moreover, the LEACH and LEACH-C protocols give results better than the protocol based on the static clustering which is slightly more powerful than the MTE protocol. This protocol is based on the choice of the shortest path to the BS.

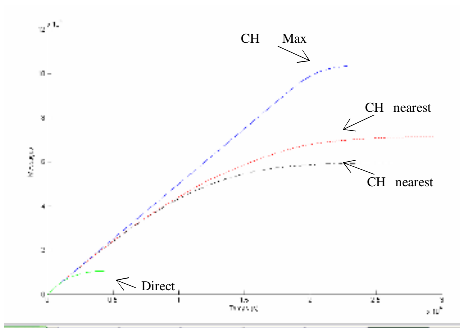


Fig. 5.c. Number of the received messages by BS VS time with LEA2C

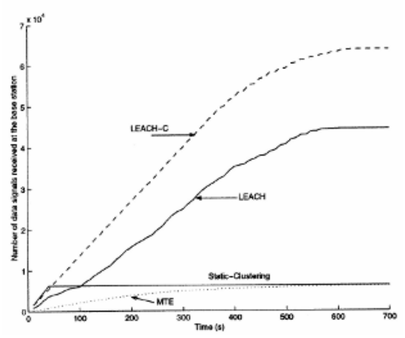


Fig. 6.c. Number of the received messages by BS VS time with LEACH [7]

By comparing the figures (5.a, 5.b and 5.c) and the figures (6.a, 6.b and 6.c), we notice that the graphs have the same paces with very apparent profits brought by our algorithm compared with the LEACH and LEACH-C protocols.

On the graphs 5.a and 5.b, we notice that LEA2C, with the better selection criterion of CHs, makes it possible to ensure a totality survival of sensors during 90% of the network lifetime. Compared to LEACH and LEACH-C, our algorithm offers a profit of 30 times more important in term of total network lifetime, and 40 times more when the CH retained is the nearest sensor to the BS.

The quantity of data received by the BS is expressed according to the number of messages sent by the sensors. Figures 5.c and 6.c show that LEA2C ensures the transmission until $10x10^4$ messages (in the case of the choice of CHs having the maximum energy) and LEACH-C only $6.5x10^4$ messages, so we have a profit of 50% with our algorithm.

7 Conclusion and Prospects

We propose a LEA2C routing algorithm in an automatic evolutionary data processing sequence and unsupervised, that ensures an important profit compared with the LEACH and LEACH-C protocols. The results obtained are very promising with a lifetime up to 40 times longer. The network coverage is insured during 90% the total time of treatment and the received data quantity by the BS is increased by the BS until 50%.

To improve the results obtained, we plan the following adaptations:

- Application of a super-clustering on the CHs, and the spreading of the clustering over several levels.
- The use of other SOMs versions and the coupling with other communication protocols.
- The optimization of the parameters of the learning algorithms (SOMs, K-means).
- The integration of other parameters in the clustering process, such as the moving speed of the sensors in the case mobiles sensors.

References

1. Technology Review, "10 Emerging Technologies That Will Change the World". (http://www.technologyreview.com). (February 2003)
2. Pottie, G.J. , Kaiser, W.J. : Wireless integrated network sensors. Communications of the ACM 43 (5), pp. 551– 558. (2000)
3. Perkins, C. , Royer, E. : Ad-Hoc On-Demand Distance Vector (AODV) Routing. IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99). (1999)
4. Singh, S. and al. : Power-Aware Routing in Mobile Ad Hoc Networks. ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98). (1998)
5. Ghiasi S. and al. : Optimal Energy Aware Clustering in Sensor Networks. SENSORS Journal, Vol. 2, Issue 7, pp. 258-269, July 2002.
6. Heinzelman, W. and al. : Energy-Efficient Communication Protocol for Wireless Microsensor Networks. Sensors 2002, 2, pp. 258 –269. (2002)
7. Heinzelman, W. and al.: An Application-Specific Protocol Architecture for Wireless Microsensor Networks. IEEE Transactions on Wireless Communications, Vol. 1, No. 4, (2002)
8. Kwon, T., Gerla, M.: Clustering with Power Control. Proceedings of MILCOM '99. (1999)
9. Kawadia, V., Kumar, P., R. : Power Control and Clustering in Ad Hoc Networks. IEEE INFOCOM. (2003)
10. Murata, T. , Ishibuchi, H. : Performance evaluation of genetic algorithms for flowshop scheduling problems. IEEE Conference Evolutionary Computation, volume 2, pp. 812–817. (1994)
11. Davies, D., L., Bouldin, D., W., : A Cluster Separation Measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1(2): pp. 224-227. (1979)
12. Juha, A. , Esa, A. : Clustering of the Self-Organizing Map. IEEE Tractions On Neural Networks volume 11, n° 3, (2000)
13. Alhoniemi, E. and al. : SOM Toolbox, Copyright (C) 2000. (http://www.cis.hut.fi/projects/somtoolbox/)
14. Ettus, M. : System Capacity, Latency, and Power Consumption in Multihop-Routed SS-CDMA Wireless Networks. In RAWCON Conference , pp. 55-58. (1998)
15. Akyildiz, I., F., Su, W., Sankarasubramaniam, Y. , Cayirci, E. : A Survey on Sensor Networks. IEEE Communications Magazine, Vol. 40, No. 8, pp. 102-114, (2002).

Author Index