Radu State
Sven van der Meer
Declan O'Sullivan
Tom Pfeifer (Eds.)

LNCS 4269

Large Scale Management of Distributed Systems

17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2006
Dublin, Ireland, October 2006, Proceedings

ifip

Springer

Radu State   Sven van der Meer
Declan O'Sullivan   Tom Pfeifer (Eds.)

Large Scale Management of Distributed Systems

17th IFIP/IEEE International Workshop
on Distributed Systems: Operations and Management, DSOM 2006
Dublin, Ireland, October 23-25, 2006
Proceedings

Springer

Volume Editors

Radu State
INRIA-LORIA, Campus Scientifique
BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France
E-mail: radu.state@loria.fr

Sven van der Meer, Tom Pfeifer
Waterford Institute of Technology, Telecommunications Software & Systems Group
Cork Road, Waterford, Ireland
E-mail: vdmeer@ieee.org, t.pfeifer@computer.org

Declan O'Sullivan
Trinity College Dublin, Department of Computer Science
Dublin 2, Ireland
E-mail: declan.osullivan@cs.tcd.ie

Library of Congress Control Number: 2006934294

CR Subject Classification (1998): C.2.4, C.2, D.1.3, D.4.4, K.6, K.4.4

LNCS Sublibrary: SL 5 – Computer Communication Networks and Telecommunications

ISSN 0302-9743
ISBN-10 3-540-47659-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-47659-7 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11907466 06/3142 5 4 3 2 1 0

Preface

This volume presents the proceedings of the *17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2006)*, which was held in Dublin, Ireland during October 23rd to 25th, 2006. In line with its reputation as one of the pre-eminent fora for the discussion and debate of advances of distributed systems management, the 2006 iteration of DSOM brought together an international audience of researchers and practitioners from both industry and academia.

DSOM 2006 was the 17th in a series of annual workshops, and it followed the footsteps of highly successful previous meetings, the most recent of which were held in Barcelona, Spain (DSOM 2005), Davis, USA (DSOM 2004), Heidelberg, Germany (DSOM 2003), Montreal, Canada (DSOM 2002) and Nancy, France (DSOM 2001). The goal of the DSOM workshops is to bring together researchers in the areas of networks, systems and services management, from both industry and academia, to discuss recent advances and foster future growth in these fields. In contrast to the larger management symposia, such as Integrated Management (IM) and Network Operations and Management (NOMS), the DSOM workshops are organised as single-track programmes in order to stimulate interaction among participants.

Following the excellent experiences from the previous year, DSOM was for the second time co-located with several related events, namely the 9th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS 2006), the 6th IEEE International Workshop on IP Operations and Management (IPOM 2006), the 2nd IEEE/IFIP International Workshop on Autonomic Grid Networking and Management (AGNM 2006) and the 1st IEEE International Workshop on Modelling Autonomic Communications Environments (MACE 2006). All these events together formed the 2nd International Week on Management of Networks and Services (Manweek 2006).

The major theme of the DSOM 2006 workshop was the management of large scale systems. Such systems are becoming a reality, including: large sensor networks, server farms, distributed content provider networks, and IP and telecommunications networks. Scalability issues and their impact on the management plane are common among all such infrastructure, and the existing management approaches are largely inadequate for emerging large scale and complex systems. The ambitious goal of DSOM 2006 was to facilitate the sharing of a first research vision on scalable network management paradigms for large scale service and network infrastructures. Rethinking network and service management from a scalability perspective and redefining which management paradigms and approaches are adequate, were the main challenges of DSOM 2006. With many papers presented at the workshop addressing some of these challenges, there was also room for papers addressing general and hot-topics related to the management of distributed systems.

In response to the DSOM 2006 call for papers a total of 85 full paper submissions were received from 25 countries, out of which 77 were reviewed. The remaining 8 papers were incomplete or withdrawn. Some submissions came from groups affiliated with the DSOM TPC co-chairs. These papers passed through a separate review process; several anonymous accounts were created on JEMS and the remaining TPC co-chairs delegated the reviews to the wider TPC, who anonymously filled in the review.

Within the comprehensive review process carried out by the technical programme committee and additional subject area experts, 75% of the submitted papers received 4 reviews and 35% of the submitted papers received 3 reviews. All submissions were ranked based on review scores as well as the wider technical programme committee's view on their contribution and relevance to the conference. After lengthy online discussions, it was decided to accept 21 of the submissions as full papers (an acceptance rate of 25.6%). Due to their relevance and quality, we recommended 5 of the submissions as short papers, of which 4 were presented at the workshop.

The papers presented here, we believe, represent novel and interesting contributions to addressing these challenges and meeting the goal of DSOM 2006 covering the following topic areas: ontologies and networks management; security and policy based management; business and service management; complexity of service management; performance of management protocols; supporting approaches for network management and management of next generation networks and services. We believe that this collection of papers provide a valuable insight into the current state of the art in techniques for scalable management for large scale service and network infrastructures.

There are many people whose hard work and commitment were essential to the success of DSOM 2006. Foremost are the researchers who submitted papers to the conference. The overall quality of submissions this year was very high and we regret that many high quality papers had to be rejected. We would like to express out gratitude to the DSOM 2006 technical programme committee, for their advice and support through all the stages of the conference preparation. We thank all paper reviewers, in particular those outside the technical programme committee, for their uniformly thorough, fair and helpful reviews. We also thank the JEMS team, which provided the infrastructure for the paper evaluation process.

We thank our sponsors, the International Federation for Information Processing (IFIP) Working Group 6.6 on Management of Networks and Distributed Systems with technical co-sponsorship by the IEEE Communications Society, Technical Committee on Network Operations and Management (CNOM). Most of the more time-consuming practical and logistical organisation tasks for the conference were handled by the members of the Manweek 2006 Organisation Committee, and this made our jobs significantly easier, and for that we are very grateful.

Finally, we wish to acknowledge the financial support of both Science Foundation Ireland and the Manweek 2006 corporate sponsors, whose contributions were hugely instrumental in helping us run what we hope was a stimulating, rewarding and, most importantly, an enjoyable conference for all its participants.

October 2006

Radu State
Sven van der Meer
Declan O'Sullivan
DSOM 2006 TPC Co-chairs

Tom Pfeifer
Manweek 2006 Publication Chair

DSOM 2006 Organisation

Technical Programme Committee Co-chairs

Radu State	INRIA-LORIA, France
Sven van der Meer	Waterford Institute of Technology, Ireland
Declan O'Sullivan	Trinity College Dublin, Ireland

Organisation Co-chairs

Brendan Jennings	Waterford Institute of Technology, Ireland
Sven van der Meer	Waterford Institute of Technology, Ireland

Publication Chair

Tom Pfeifer	Waterford Institute of Technology, Ireland

Publicity Co-chairs

Sasitharan Balasubramaniam	Waterford Institute of Technology, Ireland
John Murphy	University College Dublin, Ireland

Treasurer

Mícheál Ó Foghlú	Waterford Institute of Technology, Ireland

Local Arrangements

Miguel Ponce de León	Waterford Institute of Technology, Ireland
Dave Lewis	Trinity College Dublin, Ireland
Dirk Pesch	Cork Institute of Technology, Ireland
Gabriel-Miro Muntean	Dublin City University, Ireland
Seán Murphy	University College Dublin, Ireland
Rob Brennan	Ericsson, Ireland

Manweek 2006 General Co-chairs

William Donnelly	Waterford Institute of Technology, Ireland
John Strassner	Motorola Labs, USA

Manweek 2006 Advisors

Raouf Boutaba	University of Waterloo, Canada
Joan Serrat	Universitat Politècnica de Catalunya, Spain

DSOM 2006 Technical Programme Committee

Ehab Al-Shaer	DePaul University, USA
Aidan Boran	Bell Labs, Ireland
Raouf Boutaba	University of Waterloo, Canada
Nevil Brownlee	University of Auckland, New Zealand
Marcus Brunner	NEC Europe, Germany
Mark Burgess	University College Oslo, Norway
Omar Cherkaoui	University of Quebec at Montreal, Canada
Alexander Clemm	Cisco Systems, USA
Luca Deri	ntop.org, Italy
Gabi Dreo Rodosek	Universität der Bundeswehr München, Germany
Olivier Festor	LORIA-INRIA, France
Alex Galis	University College London, UK
Yacine Ghamri-Doudane	Institut d'Informatique d'Entreprise, France
Kurt Geihs	University Kassel, Germany
Lisandro Z. Granville	Federal University of Rio Grande do Sul, Brazil
Heinz-Gerd Hegering	Ludwig-Maximilian-University Munich, Germany
Joseph L. Hellerstein	IBM T.J. Watson Research Center, USA
James Hong	POSTECH, Korea
Cynthia Hood	Illinois Institute of Technology, USA
Gabriel Jakobson	Altusys Corporation, USA
Brendan Jennings	Waterford Institute of Technology, Ireland
Alexander Keller	IBM T.J. Watson Research Center, USA
Dave Lewis	Trinity College Dublin, Ireland
Hong Li	Intel, USA
Antonio Liotta	University of Essex, UK
Emil Lupu	Imperial College London, UK
Hanan Lutfiyya	University of Western Ontario, Canada
Jean-Philippe Martin-Flatin	University of Quebec at Montreal, Canada
Saverio Niccolini	NEC Research Lab, Germany
José-Marcos Nogueira	Federal University of Minas Gerais, Brazil
George Pavlou	University of Surrey, UK
Aiko Pras	University of Twente, The Netherlands
Juergen Quittek	NEC Europe, Germany
Danny Raz	Technion, Israel
Akhil Sahai	HP Laboratories, USA

Jürgen Schönwälder	International University Bremen, Germany
Joan Serrat	Universitat Politècnica de Catalunya, Spain
Adarshpal Sethi	University of Delaware, USA
Morris Sloman	Imperial College, UK
Rolf Stadler	Royal Institute of Technology, Sweden
Burkhard Stiller	University of Zurich and ETH Zurich, Switzerland
John Strassner	Motorola Labs, USA
Joe Sventek	University of Glasgow, UK
John Vicente	Intel, USA
Vincent P. Wade	Trinity College Dublin, Ireland
Carlos Becker Westphall	Federal University of Santa Catarina, Brazil
Felix Wu	University of California at Davis, USA
Makoto Yoshida	The University of Tokyo, Japan

DSOM 2006 Additional Paper Reviewers

Constantin Adam	Royal Institute of Technology, Sweden
Sasitharan Balasubramaniam	Telecommunications Software & Systems Group, Ireland
Keara Barrett	Waterford Institute of Technology, Ireland
Claudio Bartolini	HP Labs, USA
Steffen Bleul	University of Kassel, Germany
Ray Carroll	Waterford Institute of Technology, Ireland
Ilias Chatzidrossos	Royal Institute of Technology, Sweden
Lawrence Cheng	University College London, UK
Manish Dave	Intel, USA
Steven Davy	Waterford Institute of Technology, Ireland
Alan Davy	Waterford Institute of Technology, Ireland
Rudy Deca	Université du Québec à Montréal, Canada
Kieran Delaney	Cork Institute of Technology, Ireland
Roberto Dias	CEFET-SC, Brazil
Kevin Feeney	Trinity College Dublin, Ireland
Tiago Fioreze	University of Twente, The Netherlands
Jochen Fromm	University of Kassel, Germany
Sylvain Hallé	Université du Québec à Montréal, Canada
Enric Jaén	Universitat Politècnica de Catalunya, Spain
Georgios Karagiannis	University of Twente, The Netherlands
John Keeney	Trinity College Dublin, Ireland
Fernando Koch	University of Utrecht, The Netherlands
Abdesselem Kortebi	Université de Pierre-et-Marie Curie, Paris 6, France
Elyes Lehtihet	Waterford Institute of Technology, Ireland
Ling Lin	University of Essex, UK

Table of Contents

Efficient Information Retrieval in Network Management Using Web Services

Aimilios Chourmouziadis and George Pavlou

Center of Communications and Systems Research, Department of Electronic and Physical Sciences, University of Surrey,
GU27XH Guildford, United Kingdom
{A.Chourmouziadis, G.Pavlou}@surrey.ac.uk
http://www.ee.surrey.ac.uk/CCSR/

Abstract. Web Services is an XML-based technology that has attracted significant attention for building distributed Internet services. There have also been significant efforts trying to extend it to become a unifying management technology. An all-encompassing management technology needs to support efficient information retrieval, scalable event management, transaction support for configuration management and also security. Previous technologies, such as CMIP, SNMP and CORBA have addressed these aspects poorly, partially or at a high cost. This paper proposes an approach to address efficient information retrieval in terms of both bulk and selective data transfer. In order to achieve this, services modelling management information need to be organized in a hierarchy through service association. In order to achieve service association, information metadata are defined in secondary endpoints compared to the ones where services are deployed and accessed. We have defined a language for expressing arbitrarily complex information retrieval expressions and implemented a parser at the object level that evaluates these expressions, navigates arbitrary service associations and returns the results. We demonstrate the use and usefulness of the approach in an example usage scenario.

1 Introduction

Since the introduction of the Simple Network Management Protocol (SNMP) in the early 1990's and the versions of it that followed, its wide deployment for sophisticated network management still raises a lot of concerns. In the 2002 IAB Network Management Workshop [1] it became evident that SNMP can not be used for sophisticated management since its inneficiencies limit its potential usage to relatively simple monitoring. Therefore, alternative technologies are required to meet management goals such as efficiency in information retrieval, transaction support, security and also reduced development & operational costs. Distributed object technologies and, in particular, the Common Object Request Broker Architecture (CORBA) was considered as a unifying management technology and, although it has come a long way since then, it still has serious inefficiencies. In Corba federation and bulk retrieval are not supported, filtering capabilities lack expressiveness, scalability is an issue in addition to the large agent footprint. More recently, the introduction of Web Services, coupled with the advent of maturing eXtensible Markup Language (XML) technology standards, is seen as a promising approach for faster product development, tighter system integration and robust device management [2].

Web Services (WS) is an XML technology that encompasses W3C standards such as the Simple Object Access Protocol (SOAP) [3], the Web Services Definition Language (WSDL) [4] and the Universal Discovery Description and Integration protocol (UDDI) [5]. Since all these have their CORBA equivalents [7], WS can be used for distributed network management in a similar fashion to CORBA. But can they address this goal efficiently? Researchers in [6] and [7] compared the performance of WS, CORBA and SNMP. The conclusion was that when the amount of information to be retrieved increases, so does the efficiency of WS in comparison to SNMP. Smaller amounts of data though results in higher traffic for WS. The performance of WS, in terms of coding and latency, is poor in comparison to CORBA and SNMP.

Though the measurements in [6] and [7] show that WS could only be used in management scenarios where large amounts of data need to be exchanged, this is not necessarily true. WS performance at this stage can yield ambiguous results. As discussed in [8] and [9] approaches to resolve issues such as parsing, tranport problems, compression and data serialization etc, are still immature. Moreover the support WS provide to create sophisticated requests needs also to be investigated. Emulating the behavior of SNMP's operations such as GetNext and GetBulk is not a good practice when using WS. Such practices deprive any WS-based framework from the ability to use alternative sophisticated approaches to perform operations such as complex information retrieval. Performance and capabilities are thus inhibited.

In this paper we introduce a sophisticated approach to achieve true bulk or selective information retrieval, a capability that only CMIS/P offers among all management technologies, albeit at the cost of complexity and adherence to OSI upper layers that are not used widely anymore. In comparison, SNMP has limited support for bulk retrieval, mainly due to its mapping to UDP, and has no selective retrieval capabilities. Finally, CORBA lacks explicit support for such functionality.

Since one of our goals is to provide solutions for real management information retrieval scenarios, we have used SNMP MIBs modeled as web services to which retrieval scenarios are applied. In order to facilitate information retrieval it was important to come up with a way to organize data and services in a hierarchy that allows navigation of the information being held. To do this, we came up with a scheme to associate services and define arbitrary relationships between them. This hierarchical organization allows us to employ schemes of selective or bulk retrieval. This is done by deploying a parser at the object level on the agent side that accepts requests in the form of queries from a manager expressed in a language we designed. The agent uses the parser to interpret these queries and respond to the manager with the data collected from a list of management web services the agent has access to.

The remainder of this paper is structured as follows. In section II, we provide an analysis of our system model. In section III we present details on how service association is performed and how arbitrary service relationships can be defined. Section IV discusses details about the information retrieval grammar and the parser we developed. In section V we present a usage scenario that demonstrates the use and usefulness of our approach. Finally, in section VI we present our conclusions.

2 Service Design

Since the appearance of distributed object technologies, researchers realized the importance of converting SNMP and CMIS/P management information models. This led to the establishment of the Joint Inter Domain Management taskforce (JIDM) that produced a translation between the SNMP SMI / CMIS/P GDMO and CORBA's Interface definition Language (IDL). A JIDM-like translation though of SNMP SMI to services would be problematic for the same reasons encountered with its mapping to IDL. First bulk or selective retrieval is not supported. Second, scalability problems may arise when vast amounts of dynamic entities, i.e. SMI table rows, are modeled as separate services. As such, we considered an emerging approach for semantic translation of SNMP MIB's [7]. In such a translation, there may exist service methods for returning commonly accessed groups of single instanced objects, e.g. packet counters. In addition, tabular information is accessed collectively through a method that returns a table as a list of records. Additional methods may support SNMP-like "get next" or "get N next, i.e. get bulk" functionality. This type of modeling adds support for bulk data transfer for multiple instanced objects. Still, selective retrieval is not supported.

2.1 Information Retrieval Approaches

In WS bulk and selective information retrieval operations could be performed in two ways. The first method involves performing filtering actions at the SOAP level with a tool such as the XML Path Language (XPath) or similar. Since SOAP's body is in XML, XPath can be used to selectively pick portions of it. Such an approach is not problem-free though. Initially, extra latency is added in the retrieval process because more data need to be accessed, retrieved and coded in the SOAP body. Moreover, according to views expressed in the Network Configuration protocol mailing list, XPath is a very heavy tool for filtering purposes. Even a cut down version of XPath may still be resource intensive. A second approach to address selective retrieval is to perform it within the object code that represents a WS. As such we perform filtering before formation of the SOAP body, avoiding extra latency and keeping resource usage low by binding selective retrieval to the needs of the information model.

2.2 Modeling Approach

Supporting bulk retrieval for both single instance and multiple instance entities requires a collective view of management data across all levels of the data structure. To achieve this for SNMP, every MIB is modeled as a service. A level higher from the service level an agent has a collective view of all services, organized in a hierarchy through service association. The association scheme allows for arbitrary relationships to be defined. The agent in the scheme uses a parser to decide based on string expression queries it receives, the services from which data must be retrieved. Thus the agent has both a collective view and a selective capability over the services underneath it. At object code level, every service contains single instance and multiple instance entities of SNMP data modeled as simple values and tables. Bulk retrieval is achieved by three methods with different views on data. One method has access to single instance data, the other has view on table data and the third method has view of all the object data. All methods receive string arguments that represent command line queries that are interpreted by a parser we have built which decides which data will be sent to the manager requesting them. As such, all methods have both bulk and selective access to the data in their view. In Fig. 1 the translation of the SNMP's data into services is given. Modeling information this way initially allows a collective view of information at various levels (low level of granularity). At the same time a selective capability upon any data structure (service, simple data, tables) is offered (high level of granularity). Thus, our approach not only allows us to perform selective or bulk information retrieval but also to keep the number of services representing management data low. Therefore, it tries to avoid complexity and scalability problems.

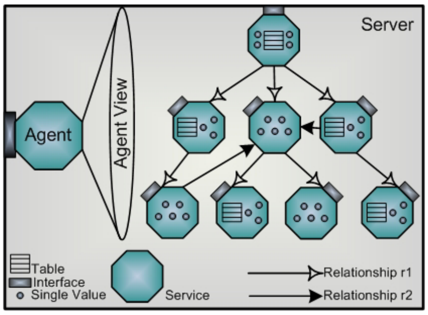


Fig. 1. Modeling approach

3 WS Association

From the information modeling approach presented in the previous section, it is evident that our scheme has two requirements. Firstly, it requires some means to define logical or physical relationships between services. These relationships will make navigation and selection of services feasible. These relationships provide an agent with a hierarchical view of the services underneath it and easy access to their data. A second requirement is that these relationships must be arbitrary so that traversal can be based on any relationship. The idea of navigation of arbitrary relationships between entities modeling management data is originated in [10]. The authors there propose to make data retrieval more efficient in CMIS/P by allowing traversal of objects based on any relationship and not only containment. Our concept is to define different relationships between services and make navigation possible based on them. Services though have different access requirements than objects (i.e. more latency). Thus, we decided that it is more efficient to make service selection first and only then to apply selective actions on the data, in a similar fashion to what CMIS/P does with scoping and filtering. With these two requirements satisfied, our agent can selectively pick up services according to string expression queries it receives from a manager.

The remainder of part III continues as follows. In section 3.1 we present the concept of navigating the relationships that are defined between services. In section 3.2 we present how to define such relationships.

3.1 Navigation and Selection of Services

The common view for a hierarchical organization of entities is that of a tree, where elements in the previous level of the tree are connected with containment relationships with the ones in the next level. Navigation among the elements in this tree is based on containment. If these elements were services capturing SNMP MIBs and the relationships between these services were arbitrary, then a tree such as the one depicted in Fig. 2 can be defined.

Navigation of this tree and selection of services by an agent is possible in our scheme, by defining simple expressions that identify a starting point for the search, level restrictions for service selection and relationship patterns to follow. A simplified Backus Naur Form (BNF) [11] syntax for such an expression is the following:

<path_selection_exp>::={< startpoint_tag> , <minlevel_tag> , <max-level_tag> , <pattern_tag>} . (1)

<pattern_tag>::=<identifier> | <pattern_tag> **.** <identifier> | (<pattern_tag>)! . (2)

<min_level_tag>::=<integer> . (3)

<max_level_tag>::=<integer> . (4)

<startpoint_tag>::=<identifier> . (5)

The path selection expression is dispatched from a manager to an agent in the form of a “command line” query expression. The agent uses it to select services based on relationship patterns, the level restrictions and the starting point tag. Selective retrieval of data from these services is performed only after selection of services has been completed. This is achieved by using the parser developed to interpret several other expressions that we will present later on. In order to demonstrate the usage of the path selection expression, some examples are given.

1) Path selection expression with no Restriction: A path selection expression such as the one in equation 6, if used with the information tree of services shown in Fig. 2, will cause a number of actions. Upon receiving the expression, the agent will use the parser to evaluate its validity. If the expression is valid then the agent will evaluate the expression and will start searching the sub-tree defined from the starting point (Root in this case) for services which can be reached by following relationships first of type r1 and then of type r2. The services selected are highlighted in Fig. 3. If selective retrieval expressions are also dispatched, the agent will only return the values that match the criteria posed by these expressions, as we will see later.

$$path\_sel\_\exp = \{Root,,,r1.r2\} \tag{6}$$

2) Path selection expression with single level Restriction: For the path selection expression in equation 7 the agent will start searching the sub-tree defined from the starting point (Root) for services in level 2 to which you can reach following relationships first of type r1 and then r2 . The selected services are highlighted in Fig. 4

$$path\_sel\_\exp = \{Root, 2, 2, r1.r2\} \tag{7}$$

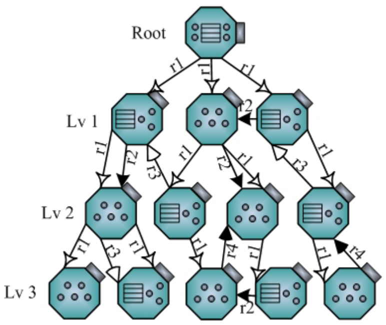


Fig. 2. General relationship tree

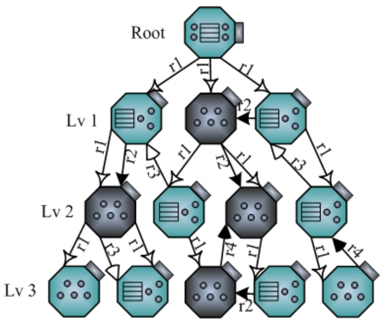


Fig. 3. Service selection no Restriction

3) Path selection expression with multiple level Restriction: In the case where the path selection expression has multi-level restriction, as in equation 8, the agent will search the sub-tree defined from the starting point (Root) for services in level 2 and 3 which can be reached by first following relationships of type r1 and then r2. The selected services are highlighted in Fig. 5

$$path\_sel\_\exp = \{Root, 2, 3, r1.r2\} \tag{8}$$

4) Fringe Services: In all the above service selection examples the agent visits one after the other all services included in the sub-tree whose head node is the start node tag. For every selection the agent makes, it evaluates for every service node whether each pattern tag in the relation pattern can be followed or not, thus there is a recursive evaluation of the binary state of each pattern tag in the relation pattern. The recursive evaluation of each pattern tag on a sequence of pattern tags can allow detection of services where the relation pattern cannot be followed. These services are called

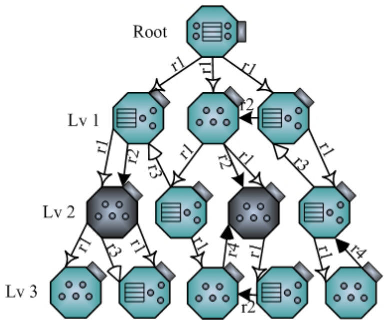


Fig. 4. Service selection single restriction

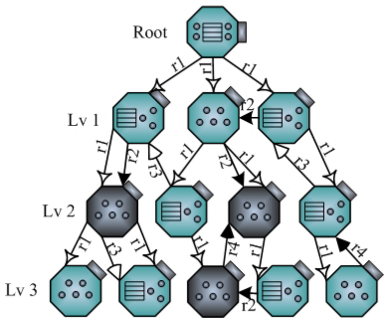


Fig. 5. Service selection multiple restriction

fringe services. An example of a path selection expression that captures services where type r1 relationships cannot be followed is given in equation 9. The services that are selected are highlighted in Fig. 6

$$path\_sel\_\exp = \{Root,,,(r1)!\} \tag{9}$$

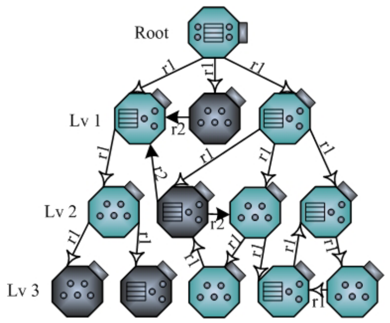


Fig. 6. Service selection for fringe Services

3.4 Service Association

In order to use the selection scheme described previously, a method to define relations between services is required. A simple scheme to define such relationships if containment was the only option would be to use a simple naming scheme to define the endpoint URIs where services are deployed. Such a scheme would be to use the number of slashes "/" in the endpoint URI to denote the level in a hierarchy where a service is offered, the tag after the last slash to denote the name of the service and the tag before it to denote its parent. However, relationships between data may not only be containment. The definition of other relations must also be possible, so that the definition of actions between data and conversation scenarios between services is feasible.

One way to define relationships between services is to provide metadata about them. Initially we considered certain WS standards for the job. WS-Addressing [14] and WS-MetadataExchange (WS-MEX) [15] are such standards. WS-MEX specifies the messages that applications exchange in order to retrieve service metadata. WS-MEX is intended though as a retrieval mechanism for only service description data. Because introduction of metadata services will increase unnecessarily latency and memory requirements WS-Addressing was considered as another solution. WS-Addressing was initially designed to provide service endpoints with capabilities such as multi protocol and interface information support. This is achieved by adding service endpoint references (ER) to endpoint's Uniform Resource Identifiers (URIs). ERs are adding information to WS messages so that applications can support various exchange patterns than simple request responses. WS-addressing allows notification standards to provide asynchronous event reporting and subscription. It can be used though in a different way. WS-Addressing could be used to add information about service relationships. The addition of a metadata element in the standard to provide a consuming application with WSDL information also allows the provision of other metadata about a service. Still work on WS-Addressing is not finalized, while the standard and the metadata element is not supported by many open source toolkits. Thus, we had to find other means to support metadata information about service relationships until work in WS-addressing is finalized and open source toolkits support it.

Providing metadata about service relationships requires a simple and flexible scheme. In WS, WSDL allows to define the interface that services expose and the access point where they are offered. A WSDL document consists of an abstract part acting as an access stub and a concrete part affecting its behavior. The interface tag of the abstract part describes sequences of messages a service sends and/or receives. It does this by grouping related messages into operations. An operation is a sequence of input and output messages, and an interface is a set of operations [4]. The endpoint component of the concrete part defines the particulars of a specific endpoint where a service is accessed [4]. A service consists of its instance and its endpoint and the later is as important as the former. Most service properties and settings reside there.

The organization of WSDL and the structure it enforces on its constituent parts allows us to do three things. First, it allows manipulation of the level of transparency [12]. Secondly, the granularity of services can be altered [12]. Third it permits three distinct ways to deploy services. The most common way of deployment is by allowing access to an entire object through an interface. Service WS0 in Fig. 7 shows this deployment scenario. The WSDL document in this case contains one service tag referring to one binding and one endpoint tag. The second deployment scenario seen in Fig. 7 is for service WS1 where access to a service is through multiple access points. In this case the WSDL document for this service contains one service tag which includes multiple endpoint tags (two in this case) referring to the same binding tag. The third deployment scenario is seen for services WS2 and WS3. In this scenario two interfaces to the same object are offered by defining multiple endpoint elements for the same service element, which refer to different binding elements.

For our association scheme use of the second deployment scheme was made. The multiple access points of this scheme provide us with the means to define metadata about service relationships. In our proposal, every service has a primary access point to provide access to it. For every relationship a service has with another service, the latter will define a secondary URI. This secondary URI provides metadata about the relationship that the two services share with a syntax that complies with RFC 3986 [13] about URIs. The syntax for the primary and the secondary URIs is given in (10) and (11). Parsing secondary URIs provides agents with a view of the association tree.

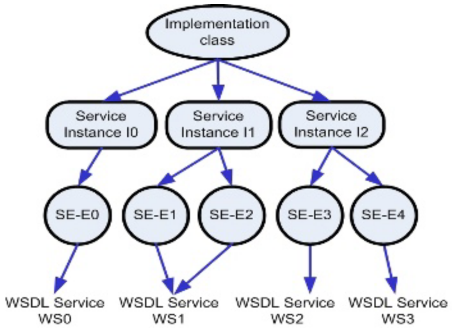


Fig. 7. Service deployment scenarios

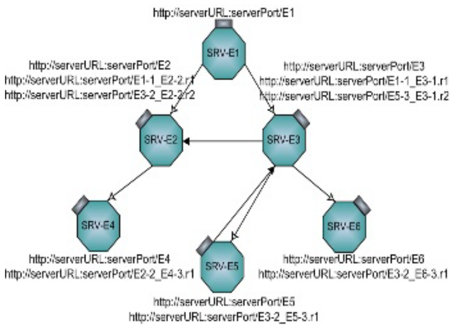

http://serverURL:serverPort/E1
SRV-E1
http://serverURL:serverPort/E2
http://serverURL:serverPort/E1-1_E2-2.r1
http://serverURL:serverPort/E3-2_E2-2.r2
http://serverURL:serverPort/E3
http://serverURL:serverPort/E1-1_E3-1.r1
http://serverURL:serverPort/E5-3_E3-1.r2
SRV-E2
SRV-E3
SRV-E4
SRV-E6
SRV-E5
http://serverURL:serverPort/E4
http://serverURL:serverPort/E2-2_E4-3.r1
http://serverURL:serverPort/E6
http://serverURL:serverPort/E3-2_E6-3.r1
http://serverURL:serverPort/E5
http://serverURL:serverPort/E3-2_E5-3.r1

Fig. 8. Association scenario with endpoints

In Fig. 8 primary and secondary URIs for all services sharing two types of relations are provided. Service ST-E1 has only one URI to allow access to it. Services ST-E2 and ST-E3 have 3 URIs, the primary and two secondary ones for both the r1 and r2 types of relations they share with other services. All third level services contain one primary one secondary URI to show an association of type r1 with other services.

Primary_URI =http://serverURL:serverPort/primaryServiceTag (10)

Secondary_URI=http://serverURL:serverPort/sendingServiceTag-serviceLevel_recipientServiceTag-serviceLevel.relationTag (11)

4 Selective Retrieval at Service Level

So far we have explained how bulk data retrieval is possible by providing a collective view upon data accessible by services at the agent level through service association and at the service level through specific methods that have collective access to the data. Path selection expressions allow our agent to deploy a selective capability on the services modeling the management information. To add filtering as selective retrieval capability on the management data within a service, our parser also evaluates data selection expressions sent to it by the manager. These expressions mandate which information from the service should be selected by the agent.

4.1 Data Selection Expressions

SNMP contains two types of data, single and multiple instance (tabular) ones. The BNF syntax for the data selection expression for single instance data is the following:

<sgl_slct_exp>::={<identifier > | <sgl_slct_exp >, <identifier> } . (12)

An example expression for retrieving the ipIndelivers, tcpOutSegs and the tcpInSegs from the TCP and IP MIB would be the following:

sgl_slct_exp={tcpInSegs,tcpOutSegs,ipInDelivers} . (13)

Retrieving multiple instance entities is a bit more complex since it requires an expression to define which entities need be retrieved and a filtering expression to retrieve only part of the data that meet specific criteria. The BNF syntax for the multi-instance selection expression and the filter expression is the following:

<mult_slct_exp>::={<mult_inst_tag> | <mult_slct_exp>, <mult_inst_tag >} . (14)

<mult_inst_tag>::=<identifier>([] | [<integer>-<integer>] | [< integer>] | [< integer>(< | >)><letter>(> |<)< integer>]) . (15)

<flt_exp>::={<mult_inst_tag ><relational operator> <value>| <flt_exp > <space><logical_operator><space> <flt_exp>} (16)

<value>::=<integer>|<string> (17)

A usage example for retrieving all TCP connections is given in (18).

ml_slct_exp={tcpConnEntry[ ]} (18)

The filter expression for retrieving only FTP and HTTP connections is given in (19).

flt_exp= {tcpConnLocalPort = 22 OR tcpConnLocalPort =80} (19)

5 Usage Scenario

To demonstrate a case where fairly complex network management data must be retrieved, we present a use case scenario. In the configuration of Fig. 9, consisting of five IP routers and 6 Local Area Networks (LAN), LAN N2 receives substantial traffic from an HTTP server it hosts. Both N2 and R1 are able to cope with this traffic. At some point though, a user in LAN4 creates more traffic by transferring large files from an FTP server in LAN3. As a result R1 and LAN2 exceed their handling capacity. The cause of congestion in router R1 must be detected by the Central Management System (CMS) responsible for managing the overall network.

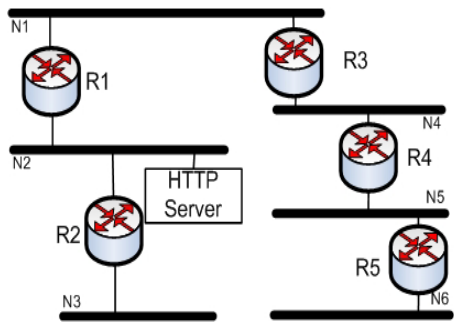


Fig. 9. LAN configuration

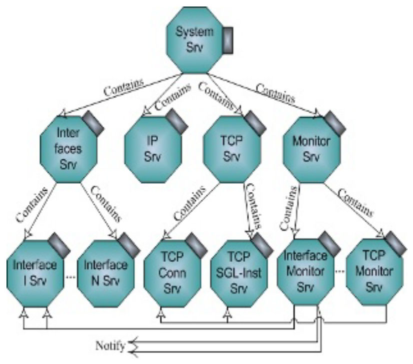


Fig. 10. Service association for usage scenario

Assuming the tree of services in Fig. 10 deployed in every host and router on the network, the CMS should be notified when congestion builds up and take appropriate actions. On notification the CMS must determine the cause of congestion and take actions to alleviate it. In Fig. 10, services IP, TCP, and Interfaces model the corresponding SNMP MIB groups. Other MIBs may also be captured as services but they are omitted. The TCP Service contains two services that model TCP single instance data and TCP connections. The Interfaces service breaks into a number of services representing the interface MIB data of every interface of the managed device. In Fig. 10, other generic services such as an event service or a logging service might also be present. To keep things simple we have omitted a WS-based notification service and assume that notifications to the CMS are sent from the services that produce them. In the future we will investigate creation of a flexible and scalable WS-based notification service. At the moment when the traffic congestion threshold is crossed on interface N2 the relevant interface monitoring service notifies the CMS; we assume here that the CMS has activated the interface monitoring service in R1 Upon receiving this notification, the CMS must try to determine the cause of congestion.

The CMS should retrieve information about the load that TCP connections between the various subnetworks are imposing on the interface N2 of router R1. Such information is not provided by SNMP MIBs. Such an option would be available, if MIBs would be capable to capture incoming and outgoing traffic per TCP connection or use the RMON MIB to capture link layer traffic. In our case, let's assume that the TCP MIB is equipped with such data in the TCP connection table under two variables called tcpConnInSegs and tcpConnOutSegs. In this case, the CMS should start monitoring hosts with Web or FTP servers and try to identify TCP connections with high segment counters and also high segment rates.

Path_slct_exp={TCP Monitor Srv,NULL,NULL,empty} (20)

flt_exp={ (tcpConnLocalPort = 22 OR tcpConnLocalPort =80) AND (21)
(tcpConnInSegs>thres_value OR tcpConnOutSegs>thres_value)

For the CMS to inform the agent which data it requires to retrieve from the TCP monitoring service, it should dispatch to the agent the expressions in (20) and (21). On receiving these two expressions, the agent picks up from the TCP monitoring service only TCP connections for applications on well known ports, usually known to produce traffic. Such applications are FTP and HTTP on ports 22 and 80 respectively. In addition, the filter expression tells the agent to only retrieve connections whose incoming or outgoing traffic exceeds a certain threshold. This way the manager will acquire a few possible candidates responsible for creating congestion. With further monitoring on these connections, it can determine the behavior of each one in terms of segment rate and possibly identify remote hosts through tcpConnRemAddress that cause the extra traffic. Without the functionality we support, the whole of the remote TCP connection table would need to be retrieved, and this would incur a lot of additional traffic to the already congested network. This is a relatively simple scenario but other scenarios also exist where a lot more information that belongs to different services must be selectively retrieved. One such case may be the selective retrieval of data from the logging service in order to trace particular series of events.

6 Conclusion

This paper presents an approach to address efficient information retrieval using Web Services. Viewing management information collectively at various levels of the management hierarchy addresses the problem of bulk retrieval. Using a parser to interpret expressions that highlight only specific data to be retrieved solves the problem of selective retrieval. The collective and selective capability we provide in our approach allows manipulation of management information at a fine level of granularity.

In order to support collective view of data at the object level, we deployed methods that view SNMP single instance or multiple instance data as a whole. To do the same at the service level we developed a scheme that defines arbitrary relationships between services. In order to allow the navigation and selection of services based on any relationship we developed a parser that interprets appropriate expressions. In order to perform selective retrieval at the object level, the same parser interprets another series expressions to highlight the data that will be selected.

We have developed both the parser which performs the path and data selection expression interpretation and the service association usage scenario to test its applicability. This was all done in Java 1.4.2.10 using the regex machine it provides. We used two WS toolkits to deploy these services, Apache AXIS and WASP, and we also plan to provide performance data in the future. Until now we have only used custom-made events that are dispatched directly from the services that produce them to the same "hardwired" manager. We will be investigating a proper notification service in the near future, tracking also work that has taken place in relevant standards bodies.

Acknowledgements

The research work in this paper was partly supported by the EU EMANICS Network of Excellence on the Management of Next Generation Networks (IST-026854).

References

1. J. Schönwälder, "Overview of the 2002 IAB Network Management Workshop," RFC 3535.
2. L. E. Menten, Bell Laboratories, "Experiences in the application of XML for Device Management," *IEEE Communications Magazine,* Vol. 42, no. 7, pp. 92-100 July 2004.
3. W3C, "The Simple Object Access Protocol 1.2 (SOAP)", http://www.w3.org/TR/soap12-part1,h ttp://www.w3.org/TR/soap12-part2
4. W3C,"The Web Services Description Language 1.1 (WSDL)", http://www.w3.org/ TR/ wsdl/
5. OASIS, "The Universal Discovery Description and Integration Technical Committee Draft V 3.02 (UDDI)," http://uddi.org/pubs/uddi_v3.html.
6. A. Pras, et al, "Comparing the Performance of SNMP and WS-Based Management," *IEEE Electronic Transaction on Network and Service Management,* eTNSM, FALL 2004.
7. G. Pavlou, P. Flegkas, and S. Gouveris, "On Management Technologies and the Potential of Web Services," *IEEE Communications Magazine,* Vol. 42, no. 7, pp. 58-66 July 2004.
8. D. Davis, M. Parashar, "Latency Performance of SOAP Implementations," 2nd *IEEE ACM International Symposium on Cluster Computing and the Grid,* 2002, p 407.
9. R. van Engelen, "Code Generation Techniques for Developing Light-Weight XML WS for embedded devices," ACM symposium on applied computing, 2004, pp.854-861
10. G. Pavlou et al, "CMIS/P++: extensions to CMIS/P for increased expressiveness and efficiency in the manipulation of management information.," 7th Annual joint conference of the IEEE Computer and Comms Societies, INFOCOM 98, Vol 2, April 1998 ,pp.430 - 438
11. P. Naur, "Revised Report on the Algorithmic Language of ALGOL 60," Communications of the ACM, May 1960, pp. 299-314.
12. J. Schonwalder, A. Pras, and J. P. Martin-Flatin, "On the Future of Internet Management Technologies," *IEEE Communications Magazine.*, Oct. 2003, pp. 90–97.
13. T. Berners-Lee, R. Fielding, L. Masinter, "The Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005.
14. D. Box, F. Curbera, "WS-Addressing specification" , W3C submission 10 August 2004.
15. F. Curbera, J. Schlimmer, "WS-Metadata Exchange specification", Sept. 2004.

On Delays in Management Frameworks: Metrics, Models and Analysis[*]

Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor

LORIA - INRIA Lorraine - Université de Nancy 2
615 rue du Jardin Botanique
F-54602 Villers-lès-Nancy, France
{Abdelkader.Lahmadi, Laurent.Andrey, Olivier.Festor}@loria.fr

Abstract. Management performance evaluation means assessment of scalability, complexity, accuracy, throughput, delays and resources consumptions. In this paper, we focus on the evaluation of management frameworks delays through a set of specific metrics. We investigate the statistical properties of these metrics when the number of management nodes increases. We show that management delays measured at the application level are statistically modeled by distributions with heavy tails, especially the Weibull distribution. Given that delays can substantially degrade the capacity of management algorithms to react and resolve problems it is useful to get a finer model to describe them. We suggest the Weibull distribution as a model of delays for the analysis and simulations of such algorithms.

Keywords: Management delays analysis, Management delays metrics, Management delays modelling.

1 Introduction

The objectives of a management framework is to maintain the service level objectives of managed systems by detecting, tracking and resolving problems that might occur on those systems. A key performance metric in management frameworks to meet these objectives is the feeding delay of management tasks and its respective quality as perceived by the management algorithms. These algorithms are susceptible to large number of managed systems with complex managed services.

Many approaches have been proposed for management systems, that varies from data-oriented approaches (SNMPv2, RMON), object-oriented approaches (Corba, JMX), mobile agents approaches [1], algorithmic approaches [2,3] and more recently autonomic management and control theory approaches [4]. Those approaches aim at minimizing the management overhead and costs but still accomplishing their management tasks with their desired goals. However, to consider a management system as efficient does not only mean reducing its cost and overhead, but also implies that the achievement of a management task has a certain level of quality, more specifically timeliness and temporal accuracy. The timeliness and the temporal accuracy aspects require that management

[*] This paper was supported in part by the IST-EMANICS Network of Excellence project.

tasks and the retrieved or altered management data must be taken and fitted within a reasonable time window [1]. Thus, one fundamental performance metric of a management algorithm is the end-to-end delay required to retrieve/alter a management attribute within a group of management nodes involved in a management task. End-to-end delays can be expressed in terms of fixed and variable components [5]. In a management system, variable delays occur due to processing time at the management nodes (managers, mid-level-managers and agents), transmission time on the network, and queueing time while the network is busy. In this study we investigate the random nature of these delays and its effect on the performance of a monitoring algorithm on the manager side, more specifically the monitoring error. Previous studies of management frameworks delays led to the establishment of simple analytical descriptions [6] or measurement based analysis where management delays are described by their two first moments (mean and standard deviation). However, we could not find a real investigation on the statistical properties of these delays.

We have developed a benchmarking platform for JMX based management applications [7] to collect management performance metrics at the application-level. In this paper, we present our results towards defining adequate models to describe management frameworks delays. Our analysis is done through a single manager/many agents configuration using a polling scheme with a fixed monitoring interval. Thereby, the analysis of the polling is interesting since it remains the main way in which GetAttribute calls occur in a managed network. We analyse the effect of the number of management agents, presented as a scalability factor, on the properties of monitoring delays. We demonstrate that there is a statistical analysis methodology and modelling for describing management frameworks performance metrics. This statistical analysis resembles in many ways to the engineering reliability modelling using the Weibull model [8] and the IP protocol performance models described in [9].

The remainder of the paper is structured as follows: Section 2 reviews related works. Section 3 describes our metrics for end-to-end management delays, and the factors that affect it. Section 4 describes our measurement methodology and data sets. In section 5, we analyse statistical properties of the delays on a polling-based monitoring algorithm. Section 6 contains a summary of this contribution as well as an outlook for future work.

2 Related Works

Many investigations have studied partially management delays as part of their proposed management frameworks, but to our knowledge none of them did propose a delay models allowing demonstrating the timeliness of management operations. Such models need the matching of delays with well known underlying statistical distributions. In literature, the main used statistics to summarize management delays was the mean and the standard deviation [10,11]. In addition, many works assume that management delays and specifically monitoring delays are uniform [12]. Such assumption is heavy and becomes invalid when the number of managed resources and management agents increase [6]. In such cases, monitoring algorithms do not scale well from a delay perspective where algorithms lost the desired quality (timeliness and temporal accuracy) when achieving tasks. This aspect of delays scaling or timeliness of management tasks has been take up by Liotta et al [1] for mobile agents based management applications as an important quality parameter for critical management tasks. Chen et al [6] assume in their work of management approaches evaluation that management delays have a non uniform random variable component that fluctuates according to the size of the managed network, but no additional study has been done to describe this non uniformness.

3 Management Delays Metrics

To define management delay metrics, we start from the IPPM framework [9] for IP packets delays measurements. We identify a monitoring delay metric to be relative to a monitoring attribute defined by its name, its delay metric type, the management operation type used to achieve the management task, the agents identifier and the number of management objects carried by the operation. For example, the delay to retrieve the *FreeMemory* attribute from a single agent to a manager is represented as following: *FreeMemory*-AttributeDelay-*GetAttribute*-Manager-Agent-1.

We identify two types of delay metrics: the attribute delay and the group delay. The attribute delay is relative to a single monitoring attribute retrieved from a single agent. This metric is for interest to capture the delay to move an attribute value between two management nodes (for example from an agent to a manager). The group delay captures the delay to move an attribute value from a group of agents with a fixed size. This metric is sensitive to perturbation that might arise on retrieving one attribute or computing an aggregated attribute from a group of agents.

The attribute delay is the total delay that an attribute experiences to retrieve/alter its values with a specific interaction mode (polling or notification), using a specific management operation. The attribute delay is measured in a time unit (seconds or milliseconds) per target attribute. If the interaction mode is notification, then it is a one way delay experienced from a sender to a receiver. If the interaction mode is polling, it is the round trip time between two or more management nodes.

The group delay is the amount of time required for a management algorithm to retrieve/alter the value of an attribute from a group of agents. The group delay is indeed the longest delay served, during a polling round, from any agent of the group. The unit of measure of the group delay is time units per agents group size. Many management algorithms are sensitive to this amount of time since it impacts the algorithm reaction time. During a management task period and according to its scheduling operations approach (sequential/serial or concurrent/parallel) [13], the group delay is defined by the maximum of attribute delays from the agent's group in a parallel management operations scheduling approach, or the sum of the attribute delays in a serial approach.

4 Measurement Methodology

4.1 Application-Level Versus Packet-Level Measurement

There are three approaches to delay measurement [14]. One is to collect packet-level information somewhere in a network path between the manager and agents. Another is to collect kernel-level information on the manager side. The third is to collect information at the application level on a manager or a mid-level managers side. The packet and the kernel level approaches provide the most detailed information and the most accurate timings. Application-level measurements are easy to implement, and the benchmarking tools are portable. We use application-level measurement to develop and evaluate management delays. The benefit of application-level measurement, in contrast to other levels measurement, is that it reflects delays as perceived by a decision point implemented on a manager or a mid-level manager sides. Thus, the measured delays will include network delays, requests and responses processing delays on each management node, matching and searching attributes values delays on the agent, and security and privacy processing. Figure 1 depicts a time series of measured delays on the manager side at the

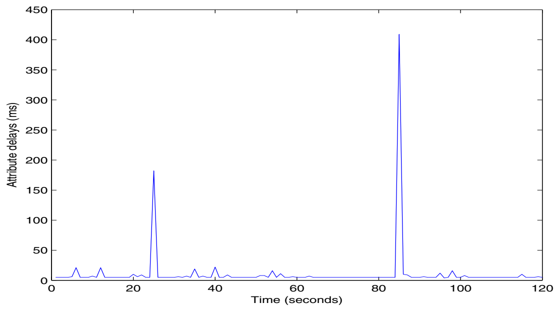


Fig. 1. A time series of measured delays at the application-level on a manager side while sending 1 request per second to a single agent over JMX framework

application-level in a Java-based management framework. We observe some periodic spikes where delay increases due to garbage collector activities each 60 seconds. Therefore, only an application-level measurement technique captures easily these spikes that might affect the performance of the management algorithm. The disadvantages of the application-level measurement are the overhead that introduces when logging measurements datasets, and the loss of information about each individual component delays among agent delays, manager delays and messages delays.

4.2 Data Collection

The data we present and analyse in this paper was collected from our JMX benchmarking platform [15] running on a cluster of 100PC (I-Cluster2) [1] where nodes are connected via a gigabyte Ethernet. We built a synthetic benchmarking application based on some widely used JMX implementations (SUN Reference implementation 1.2 and MX4J 2.0). The synthetic application is used because it provides a flexibility in experimentation with various levels of workload: number of requests per second, type of requests, type of MBeans and the number of attributes on each registered MBean within the MBean server. We used a BEA WebLogic JRockit JVM from a JDK 1.4.2_04 running on an Itanium 2 with 2x900MHZ CPU and 3GB memory. We kept the default

[1] http://i-cluster2.imag.fr

options values of all running JVMs on nodes. For all experiments we fixed the monitoring rate in terms of number of requests per second and we varied the number of agents. On the manager side, we used a concurrent monitoring operations scheduling strategy. The manager creates for each agent a pool of threads that generates a fixed number of requests per second. We use the JMX *getAttribute* operation that carries out a single attribute value from a single MBean. Thus, the size of requests is the same for all experiments. For each agent we stored its requests delays in a separate file, during a measurement of 20 minutes after a warm-up period of 1 minute as recommended in [16]. Within the measurement period, we record the timestamps before and after calling the *getAttribute* operation on the manager side. The difference between the two timestamps represents the measured attribute delay. We use the Java instruction *current-Timemillis* to get the current time in milliseconds with a resolution of 1 ms on Linux operating system[2]. All measurement data are collected and stored on a separate node. Confronted with 20GB of collected data to analyse, it is clear that we cannot hope to individually analyse each trace. We must indeed turn to *automated analysis*. That is, we use developed Perl scripts to analyse our data and generate the corresponding statistical properties. For the statistical analysis of delays, we are refereed to the same methodology proposed by Paxon in his PhD thesis [17] on which the IPPM framework lies [9].

4.3 Statistical Analysis Techniques

The first analysis technique we applied is summarizing a data set. If we wish to summarize the attribute delays we might at first think to express them in terms of their sample mean and variance (or standard deviation). However, in practice we find that often the collect of an attribute experiences one or more delays that are much higher than the reminder. These extreme delays greatly skew the sample mean and the variance, so that the resulting summaries do not accurately describe a typical behavior. A typical management delays description is reflected by the median and the IQR (Inter-Quantile-Range) that remain resilient in the presence of extremes or outliers. This *typical description* is suitable for management algorithms that require unbounded delays to retrieve management data. However, an *extreme description* is reflected by the mean and the standard deviation since they are less robust to outliers. Hence, the choice of statistical estimators for summarizing delays datasets depends on the evaluated management algorithm quality requirements. For a real time monitoring algorithm, the mean and the standard deviation are more adequate to summarize monitoring delays since they reflect any perturbations that might occur on the monitoring system. One other technique we apply, is describing delays using statistical distributions of collected data. The *empirical distribution function* (EDF) [9] of a set of scalar measurements is a function $F(x)$ which for any x gives the fractional proportion of the total measurements that were less equal than x. If x is less than the minimum value observed, then $F(x)$ is 0. If it is greater or equal to the maximum value observed, the $F(x)$ is 1. Moreover, in our analysis we match the group and attribute delays data sets against the popular Weibull distribution

[2] The resolution or granularity of a clock is the smallest time difference that can be measured between two consecutive calls.

model [8] to approximate the underlying statistics. When using this model to approximate the empirical data, we always apply *Maximum Likelihood Estimator* (MLE) [8] for parameters estimation.

5 Statistical Properties of Monitoring Delays

5.1 Attribute Delays Analysis

Firstly, we analyse the attribute delay when transferring the values of the same attribute from a single agent within a variable group size of agents to a manager under a fixed monitoring rate of 1 request per second. We know that the performance bottleneck is located at the manager side since the management approach is centralized [6]. Table 1 shows the attribute delay statistics measured on the manager side when retrieving values of an attribute from an arbitrary agent within a group. We observe from table 1 that

Table 1. Summary of attribute delays statistics measured at the manager side of a single agent within a varied group size

Group size	Attribute Delay (ms) statistics					
	Mean	Std	Median	IQR	Min	Max
70	1.02	0.20	1	0	1	4
140	4.45	46.34	1	0	1	973
210	68.47	432.37	1	0	1	3614
280	105.08	983.41	1	0	1	9960
350	392.73	2250.64	1	0	1	20269
420	675.24	3689.65	1	0	1	33365
490	561.55	4254.14	1	0	1	39816
560	1787.82	8179.39	1	0	1	50395
630	1916.86	8764.2	1	0	1	60646
700	2394.55	12073.7	1	0	1	78994

the attribute delay description varies according to the used statistics estimators as mentioned in the section 4.3. When increasing the size of the group of agents, we observe first that the minimum delay is constant, the mean and the standard deviation increases. Indeed, the median and the IQR [3] remain invariant and constant. Figure 2 shows the time series of attribute delays of different agents from groups of size 70 and 700. We observe that the attribute delays become more randomness while the group of agents size increases. We also observe that the number of delays greater than the monitoring interval of 1 second with a group of 700 agents increases. This is due to queueing delays at the manager side that experiences monitoring calls when the number of agents increases and the manager saturates. Figure 3 shows the empirical distribution of an attribute delay from an agent within a group size of 700. We did not find a suitable statistical distribution fitting these delays. Indeed, we observe that the distribution of attribute delays from an agent is multi-modal as depicted in the figure 3. The percentile

[3] The Inter Quartile Range is the $0.75 - quantile$ minus the $0.25 - quantile$.

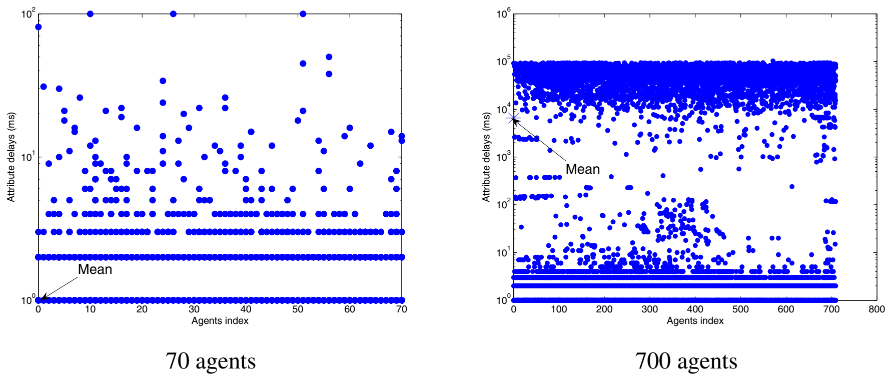


Fig. 2. Empirical time series of attribute delays per agent from two groups of agents of different sizes (y-axis is in log scale)

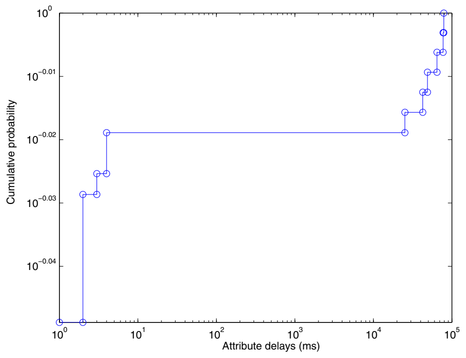


Fig. 3. Log-Log plot of the Empirical Distribution of the attribute delays of an agent within a group of 700 agents

α of attribute delays less than the monitoring interval of 1 second follow closely a normal distribution $N(1.10, 0.44)$. These delays represent monitoring calls that experience less queueing delays on the manager side. However, the $1-\alpha$ of delays greater than the monitoring interval follow closely a Weibull distribution. Despite the multi-modal behavior of the attribute delay, we claim that the attribute delays follow a weibull distribution since the normal distribution is a particular case of a weibull one. [4]

5.2 Group Delay Analysis

Secondly, we analyse the group delay of an attribute from a varied size group during a management period (a polling round for example) with a given management task (monitoring or configuration) and using a given management operation. Within a management period, the group delay is defined as a random process $X = \{X_i\}_{i=1,2,,..m}$ of a group of

[4] If the shape parameter is close to 3.602 the weibull distribution is close to a normal.

agents with size m. In this section, we are interested in the behavior of X. We plot the EDF function as depicted in figure 4 of the mean attribute delays from each agent with different group sizes varying from 70 to 700. The group mean attribute delays shift right when the group size increases that means the monitoring messages are more burstiness and the queueing delays increase on the manager side. We use the maximum likelihood

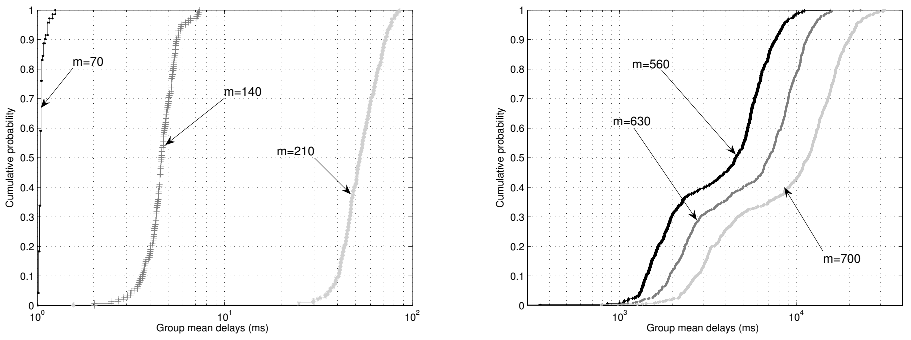
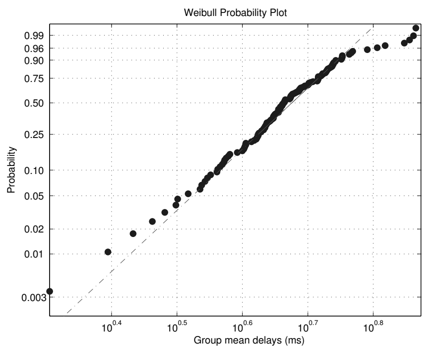


Fig. 4. Empirical Distribution of the group mean delays of a monitoring task with a monitoring period of 1 second, using a polling interaction between one manager and a group of agents of size m, and using the JMX GetAttribute operation (x-axis is in log scale)

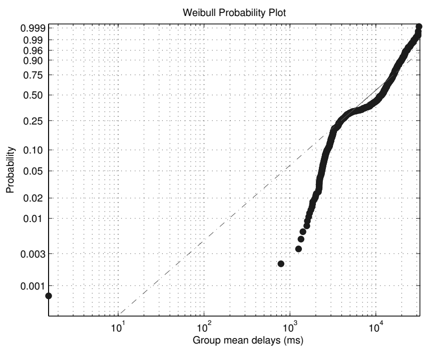
estimators [8] to find an adequate classical statistical underlying distribution of group delays with a fixed size. Our finding is that the Weibull distribution best fits the group delays data set. This distribution has been recognized as a good model for TCP inter-arrival times [18] and for many fields in engineering reliability components lifetimes [8]. Figure 5 reports the Weibull probability plots [5] of the group attribute mean delays distribution of respectively 140 and 700 agents. We find that for a size of 140 agents the group mean delays best fit the weibull distribution than a larger group of 700 agents. We find that as the group size becomes larger the group mean delays best fit a normal distribution. It is known that the normal distribution well fits sample means of a large number of independent observations from a given distribution [19][page 494]. Consequently, samples mean delays of a group of agents well fit a normal distribution as the size of the group becomes larger. The parameters a, b of the Weibull distribution represent respectively the so called *scale* and *shape* parameters which determine its structure and statistics. Varying the values of these two parameters highly impacts the appearance of the Weibull distribution. The parameter a is closely related to the distribution peak, and the parameter b is concerned with the tail behaviour.

When the shape parameter $b \gg 1$, the mode, the median and the mean are close to the value of the parameter a. Thus, parameter a represents the mean delays values where data are mainly located (distribution peak). We observe from table 2 that with a small group of 70 agents, we have a shape parameter of 17.11 large greater than 1. Therefore,

[5] The purpose of a Weibull probability plot is to graphically assess whether the data in the random variable X could come from a Weibull distribution. If the data are Weibull the plot will be linear.



140 agents



700 agents

Fig. 5. The Weibull probability plot of group attribute mean delays of group sizes of 140 and 700 agents. The fitted Weibull distributions have respectively a scale parameters $a(140) = 5.02$, $a(700) = 12349$ and a shape parameters $b(140) = 5.52$,$b(700) = 2$.

for small groups the group mean delays well fit a weibull distribution and the first and second-order statistics are close to the scale parameter. We also observe that the shape parameter decreases, as the size of the group of agents increases. When the group of agents becomes larger enough, the shape parameter is close to 2 and the Weibull distribution approximates a normal distribution. It is obviously that the normal distribution is used whenever the randomness is caused by several independent sources acting additively; for example sample means of a large number of independent observations from a given distribution.

5.3 Management Delays Quality

To gauge how well management delays scale when the size of a group of agents increases, we are interested to identify the proportion of attribute delays that are less or equal to a *maximum tolerable delay* that we consider, in this work, as the monitoring interval Δ (1 second in our case). Let θ be this proportion of attribute delays, statistically we obtain $\theta = P[X \leq \Delta] = EDF(\Delta)$. The quantity $1 - \theta$ represents the proportion of attribute delays that are late after the monitoring interval and θ is a quality parameter of a monitoring framework delay. For the group delay metric, the parameter θ becomes more interesting, since it captures the proportion of agents that respond within a delay less or equal to the monitoring interval. This quantity represents the monitoring error of the management system and captures the quality of the monitoring algorithm temporal accuracy. Table 2 shows the measured and predicted values from the fitted distribution of the $1 - \theta$ parameter.

5.4 Cross-Validation

In order, to give a first insight on the validity of our model, we use the group mean attribute delays dataset from the study of Pras et al [20] to partially verify that group delays approximate a Weibull distribution. The validation is partial because their datasets

Table 2. The measured and the predicted monitoring error $1-\theta$ from group mean delays of different sizes group of agents

Group size (m)	Measured monitoring error $1-\theta$	Predicted monitoring error $1-\theta$	Weibull parameters	
			Shape (b)	Scale (a)
70	0	0	17.11	1.07
140	0	0	5.52	5.02
210	0	0	4.54	59.16
280	0	0	1.81	77.54
350	0.05	0.04	2.28	610.36
420	0.51	0.62	1.9	1468.88
490	0.91	0.86	2.1	2519.79
560	0.99	0.95	1.8	4902.03
630	0.99	0.98	1.8	7437.27
700	0.99	0.98	2	12349

are measured for SNMP based monitoring system and are packet-level measurements. We focus on the dataset delays of retrieving a single object from the group of agents. Our finding that their group mean attribute delays of their data set with a group size of 23 agents fits closely a Weibull distribution as depicted in the figure 6. In their paper, the authors record SNMP agents Round-Trip delays at the UDP-level, indeed our dataset delay are recorded on the application-level with RMI as underlying protocol that lies on the TCP protocol. As noted in [18], wired TCP flow arrivals are well modeled by a 2-parameters Weibull distribution and claimed that it gives a better fit than other distribution models (lognormal, pareto and exponential). Hence, according to these works based on packet-level measurement and our work based on application-level measurement, we claim that the Weibull model is a good candidate to model delays in management frameworks.

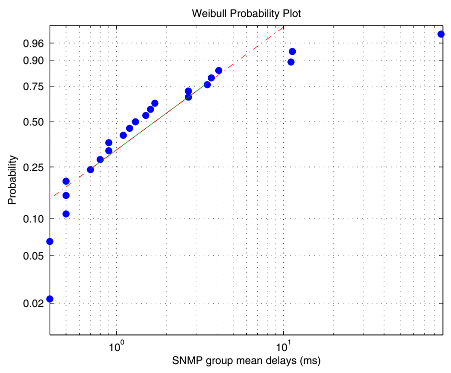


Fig. 6. The Weibull Probability plot of group attribute mean delays of SNMP agent's group size of 23. The fitted Weibull distribution has a scale parameter $a = 3.46$ and a shape parameter $b = 0.62$.

6 Conclusion and Future Works

In this paper, we have analysed management delays within a simple centralized monitoring algorithm. Our primary objective is to identify a set of metrics that characterize delays. We based our study on standardized work from other fields, specially the IPPM framework and Vern Paxon work [17]. We identified how to use the statistical estimators (mean, standard deviation or median and IQR) to characterize management delays based on the quality requirements (delays dependant or independent) of the evaluated algorithm. Our analysis of a synthetic JMX based management benchmark datasets, shows that group delays have a statistical underlying distribution, identified as the Weibull model. Understanding the statistical modelling of management delays is important for the simulation and analytical performance evaluation studies of management frameworks. It is also interesting for the proper designing of management applications (buffer sizing and underlying transport protocol). In this work we initiate an empirically derived analytical models of management frameworks, that could be exploited in simulation environments or performance prediction. As a further interesting work, we will investigate the coupling of packet-level and application-level delays measurement datasets to better understand management frameworks delays and develop more accurate models. Although the polling model is probably the main way in which Get calls occur in a managed network, it will be interesting to study how the network would behave with notifications, particularly when notifications signal errors, and this is the time that the network is most critical.

References

1. Liotta, A., Knight, G., Pavlou, G.: On the performance and scalability of decentralized monitoring using mobile agents. In: DSOM. (1999) 3–18
2. Chen, Y., Bindel, D., Song, H., Katz, R.H.: An algebraic approach to practical and scalable overlay network monitoring. In: Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 30 - September 3, 2004, Portland, Oregon, USA, ACM (2004) 55–66
3. Chua, D., Kolaczyk, E.D., Crovella, M.: A statistical framework for efficient monitoring of end-to-end network properties. In: Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2005, June 6-10, 2005, Banff, Alberta, Canada, ACM (2005) 390–391
4. Hellerstein, J.L., Diao, Y., Parekh, S., Tilbuyr, D.M.: Feedback Control of Computing Systems. Wiley-Interscience (2004) ISBN: 0-471-26637-X.
5. Bolot, J.C.: End-to-end packet delay and loss behavior in the internet. In: SIGCOMM '93: Conference proceedings on Communications architectures, protocols and applications, New York, NY, USA, ACM Press (1993) 289–298
6. Chen, T.M., liu, S.S.: A model and evaluation of distributed network management approaches. IEEE journal on selected areas in communications **20** (2002)
7. Lahmadi, A., Andrey, L., Festor, O.: On the impact of management on the performance of a managed system: A jmx-based management case study. In: DSOM. Volume 3775 of Lecture Notes in Computer Science., Springer (2005) 24–35
8. Johnson, N.L., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions. 2 edn. Volume 1. John Wiley & Sons (1994) ISBN: 0-471-58495-9.

9. Paxson, V., Almes, G., Mahdavi, J., Mathis, M.: RFC 2330: Framework for IP performance metrics (1998) Status: INFORMATIONAL.
10. Lim, K.S., Stadler, R.: Weaver: Realizing a scalable management paradigm on commodity routers. In: Integrated Network Management. Volume 246 of IFIP Conference Proceedings., Kluwer (2003) 409–424
11. Gu, Q., Marshall, A.: Network management performance analysis and scalability tests: SNMP vs. CORBA. In: IEEE/IFIP Network Operations & Management Symposium, Seoul, Korea. (2004)
12. Moghe, P., Evengelista, M.: Rap-rate adaptive polling for network management applications. In: Network Operations and Management Symposium, 1998, NOMS 98, IEEE. Volume 3. (1998) 395–399 ISBN: 0-7803-4351-4.
13. Beverly, R.: RTG: A Scalable SNMP Statistics Architecture for Service Providers. In: Proceedings of the 6th Systems Administration Conference (LISA 2002). (2002) 167–174
14. Downey, A.B.: An empirical model of tcp performance. In: MASCOTS, IEEE Computer Society (2005) 45–54
15. Andrey, L., Lahmadi, A., Delove, J.: A jmx benchmark. Technical Report RR-5598, Loria-INRIA Lorraine (2005)
16. Demarey, C., Harbonnier, G., Rouvoy, R., Merle, P.: Benchmarking the round-trip latency of various java-based middleware platforms. Studia Informatica Universalis Regular Issue **4** (2005) 7–24 ISBN: 2-912590-31-0.
17. Paxson, V.E.: Measurements and analysis of end-to-end Internet dynamics. PhD thesis, Berkeley, CA, USA (1998)
18. A.Feldmann: Characteristics of TCP connections. In: Self-similar Network Traffic and Performance Evaluation. Jonhn Wiley and Sons (2000) 367–399
19. Jain, R.: The art of Computer Systems Performance Analysis. John Wiley & Sons, Inc (1991) ISBN : 0-471-50336-3.
20. Pras, A., Drevers, T., de Meent, R.V., Quartel, D.: Comparing the performance of SNMP and web services-based management. eTransactions on Network and Service Management(eTNSM) **1** (2004)

Performance Analysis of SNMP over SSH

Vladislav Marinov and Jürgen Schönwälder

Computer Science
International University Bremen
Campus Ring 1, 28759 Bremen, Germany
{v.marinov, j.schoenwaelder}@iu-bremen.de

Abstract. There have been several attempts in the past to secure the Simple Network Management Protocol (SNMP). Version 3 of the SNMP protocol introduced a User-based Security Model (USM) which comes with its own user and key-management infrastructure. However, many operators are reluctant to introduce a new user and key management infrastructure just to secure SNMP. This paper describes how the Secure Shell (SSH) protocol can be used to secure SNMP and it provides a performance analysis of a prototype implementation which compares the performance of SNMP over SSH with other secure and insecure versions of SNMP.

1 Introduction

Network devices maintain large amounts of management data. Management data can provide insights as to how the network is performing or which abnormal events have been observed. Moreover, management data can be used to understand how a device is configured and to change the configuration of that device. The Simple Network Management Protocol (SNMP) [1] allows both for management data to be collected remotely from devices and for devices to be configured remotely. It was first published in August 1988 and since then it has been widely used in network management.

There was no security implemented in SNMP version one (SNMPv1) and the attempts to add security to SNMP version two (SNMPv2) lead to failure as well. Version 3 of the Simple Network Management Protocol (SNMPv3) added security to the previous versions of the protocol by introducing a User-based Security Model (USM) [2]. The USM was designed to be independent of other existing security infrastructures, to ensure it could function when third party authentication services were not available, such as in a broken network. As a result, USM utilizes a separate user and key management infrastructure.

Network operators have reported that deploying another user and key management infrastructure introduces significant costs and hence the USM design is actually a reason for not deploying SNMPv3. To address this issues, a new security model is currently being defined by the Integrated Security Model for SNMP (ISMS) working group of the Internet Engineering Task Force (IETF) which leverages the Secure Shell (SSH) [3] protocol. It is designed to meet the security and operational needs of network administrators, maximize usability in operational environments to achieve high deployment success and at the same time minimize implementation and deployment costs to minimize the time until deployment is possible.

This paper describes the SSH security model for SNMP and provides a performance evaluation of a prototype implementation. It is structured as follows: Section 2 describes the extensions of the SNMP architecture that are needed to support security models where security services such as authentication and encryption are provided by the message transport rather than the SNMP protocol itself. Section 3 introduces the SSH security model for SNMP. A prototype implementation of SNMP over SSH is described in Section 4 and some performance figures are presented in Section 5. Section 6 discusses related work before we conclude our paper in Section 7.

2 Extensions of the SNMP Architecture

The SNMP architecture [4] was designed to be modular in order to support future protocol extensions such as additional security models. The architecture defines several subsystems and interfaces between subsystems that should remain unchanged when subsystems are extended. The goal was to reduce side effects that can occur without such an architectural framework when the protocol is extended.

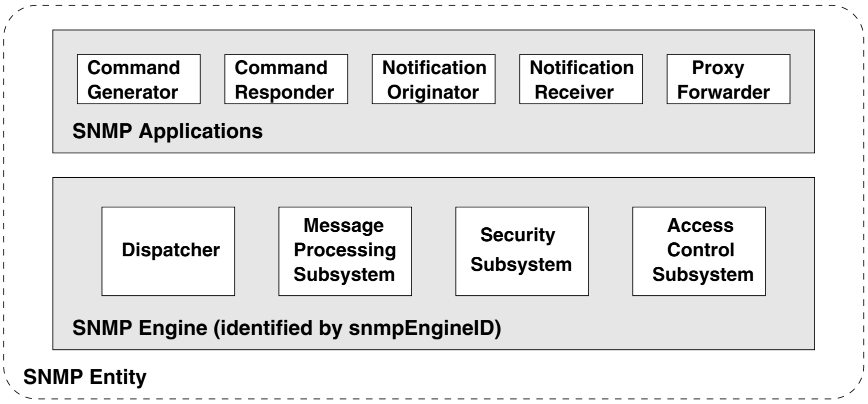


Fig. 1. Structure of an SNMP entity according to the SNMPv3 architecture

According to the SNMP architecture, an SNMP engine consists of a message processing subsystems, a security subsystem, an access control subsystem, and a single dispatcher (Fig. 1). Each subsystem can contain multiple concrete models that implement the services provided by that subsystem. The interfaces between subsystems are defined as Abstract Service Interfaces (ASIs). The dispatcher is a special component which controls the data flow from the underlying transports through the SNMP engine and up to the SNMP applications[1].

As of today, most SNMPv3 implementations support three message processing models for SNMPv1, SNMPv2c, and SNMPv3 and two security models, namely the User-based Security Model (USM) (used by the SNMPv3 message processing model) and the Community-based Security Model (CSM) (used by the SNMPv1 and SNMPv2c message processing models). There is only a single View-based Access Control Model (VACM) so far.

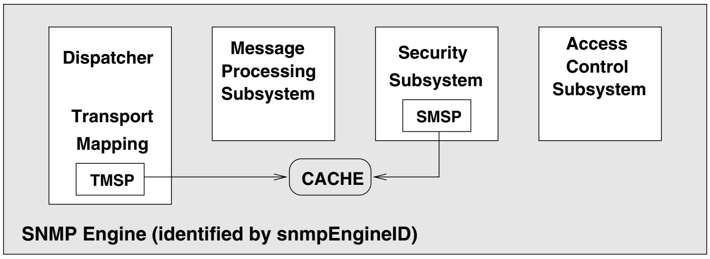


Fig. 2. Structure of an SNMP engine that supports transport mapping security models. The Transport Mapping Security Processor (TMSP) and the Security Model Security Processor (SMSP) communicate via a shared cache.

The design of the SNMP architecture assumes that security services (authentication, data integrity checking, encryption) are provided as part of the SNMP message processing. If, however, the security services are provided by the transport over which SNMP messages are exchanged, the architecture does need some extensions. The approach followed by the ISMS working group of the IETF [5] is to split a transport mapping security model (TMSM) into two parts (Fig. 2):

- The *Transport Mapping Security Processor* (TMSP) is the portion that is part of the message transport and performs the actual security processing.
- The *Security Model Security Processor* (SMSP) is the portion that realizes the appropriate security model required by the SNMPv3 architecture. In order to provide the required services, it has to interact with the TMSP.

The TMSP and the SMSP need to exchange information (e.g., the name name of the authenticated SSH user). While this exchange can be realized in different ways, the simplest and most efficient scheme is to establish a cache which maintains relevant information and to pass a handle between the TMSP and SMSP by extending the ASIs.

[1] The SNMP architecture should actually have a separate transport subsystem with proper ASIs to cleanly model the fact that SNMP supports multiple transport models. The introduction of a transport subsystem has recently been proposed to the ISMS working group.

Transport security protocols are typically session-based. They usually have a session establishment phase where a session key and some shared state is established followed by the secured data exchange. This is very different from the message-based approach used by USM where all security information is carried in every single message exchanged between two SNMP engines and there is no notion of a session or a session key.

3 SSH Security Model for SNMP

The Secure Shell (SSH) protocol [3] is a protocol for secure remote login and other secure network services over an insecure network. It consists of three major components:

– The *Transport Layer Protocol* provides server authentication, confidentiality, and integrity. It may optionally also provide compression. The transport layer protocol typically runs over a TCP/IP connection, but might also be used on top of any other reliable data stream. It uses public-key cryptography to authenticate the server to the client and to establish a secure connection which then uses a session key and a symmetric encryption algorithm to protect the connection.
– The *User Authentication Protocol* authenticates the client-side user to the server. It runs over the transport layer protocol. SSH supports several different user authentication mechanisms such as password authentication, public-key authentication, and keyboard-interactive authentication (which supports challenge-response authentication mechanisms).
– The *Connection Protocol* multiplexes the encrypted connection into several logical channels. It runs over the transport layer protocol after successful completion of the user authentication protocol. Every channel has its own credit-based flow control state in order to deal with situations where channels are connected to applications with different speeds.

Note that SSH authentication is usually asymmetric: An SSH server authenticates against an SSH client using host credentials (host keys) while the user authenticates against the SSH server using user credentials (user keys or passwords).

The SSH Security Model (SSHSM) for SNMP [6] is an instantiation of a TMSM which uses SSH, a protocol already widely deployed to secure access to command line interfaces on network elements. The specification details the elements of procedure for the TMSP and the SMSP portions of the SSHSM. It also deals with details such as `engineID` discovery and the handling of notifications. Notification delivery is not straight forward due to the asymmetric authentication provided by SSH and the requirement to exercise access control in a consistent way for read, write, and notify access. The details are still being discussed in the ISMS working group of the IETF at the time of this writing.

With the SSHSM, no security parameters are conveyed in SNMPv3 messages. Accordingly, the `msgSecurityParameters` field of SNMPv3/SSH messages carries a zero length octet string and the implementation of the security model portion of the SSHSM simply retrieves the necessary information by accessing a cache which is shared between the transport mapping porting and the security model portion of the SSHSM.

4 Implementation

The prototype implementation of SNMP over SSH developed at IUB is an extension of the widely used open source `Net-SNMP` SNMP implementation. For the SSH protocol, the `libssh` library was used, an open source C implementation of SSH. The `libssh` library contains all functions required for manipulating a client-side SSH connection and an experimental set of functions for manipulating a server-side SSH connection.

The implementation does not implement the SSHSM as it is currently discussed in the IETF since the details of the SSHSM were not worked out when the implementation work was done. The prototype only supports the TMSP part of SSHSM plus a slight modification of the Community-based Security Model (CSM) which passes the authenticated SSH user identity as the security name to the access control subsystem. This basically gives us SNMPv1/SSH and SNMPv2c/SSH while the ISMS working group defines SNMPv3/SSH.

The implementation itself consists of a new transport module which is in the order of 1200 lines of C code. The `Net-SNMP` internal API for adding transports worked well and did not require any changes. The fact that `Net-SNMP` already supports stream transports was convenient. For password authentication, the prototype calls the Linux Pluggable Authentication Modules (PAM) [7] library to make it runtime configurable how passwords are checked.

Most of the development time was spend on optimizing the performance of the implementation since the overall latency initially was surprisingly high. In order to optimize the performance of the SSH transport domain, we investigated the influence of TCP's Nagle algorithm as well as the windowing mechanism of the SSH protocol.

4.1 TCP Nagle Interactions

During our initial measurements, we observed that the execution of a `snmpget` operation over the SSH transport domain required approximately $800ms$. This surprisingly large delay was introduced by TCP's Nagle algorithm which essentially delays the sending of a TCP segment until either a segment has been filled or the previous segment has been acknowledged. We therefore disabled the Nagle algorithm by setting the `TCP_NODELAY` flag on the agent and on the manager side of the connection. This lead to a significant improvement in the performance of the SSH transport domain as the time required for the execution of a `snmpget` operation went down to $56.5ms$. We further modified the `libssh` library to disable the Nagle algorithm immediately after establishing the TCP connection between the agent and the manager and before any SSH exchanges. This further reduced the time required for a `snmpget` operation to $16.17ms$ on our fast machines.

4.2 SSH Window Adjustments

The SSH windowing mechanism is used to specify how much data the remote party can send before it must wait for the window to be adjusted. In the OpenSSH implementation such window adjustment messages are only exchanged periodically. During our initial observations we noticed that each message exchanged between the agent and the manager was followed by a window adjustment message. These additional messages introduced significant bandwidth overhead as well as latency overhead for long sessions. As a result the SSH transport domain performed worse than the USM transport domain with respect to latency and bandwidth. In order to optimize the performance, we modified the `libssh` library to send window adjustment messages only when necessary. This improvement lead to better bandwidth and latency performance of the SSH transport domain when compared to the USM transport domain as explained below.

5 Performance Analysis

The performance of our SNMP over SSH prototype has been evaluated by comparing it against SNMPv3/USM with authentication and privacy enabled, running over both TCP and UDP. In addition, to establish a baseline, the performance of plain SNMPv2c over both TCP and UDP was measured. The following sections first describe the experimental setup and then discuss the session establishment overhead and the performance for walks of different sizes without and with packet loss. Finally, the bandwidth used by the different SNMP transports is compared and the memory requirements for keeping open SSH sessions on a command responder is discussed.

5.1 Experimental Setup

The experiments were performed on three Debian GNU/Linux machines (see Table 1). The machines were connected via a switched Gigabit Ethernet with sufficient capacity.

Table 1. Machines used during the measurements

Name	**CPUs**	**RAM**	**Ethernet**	**Kernel**
`meat`	2 Xeon 3 GHz	2 GB	1 Gbps	2.6.16.14
`veggie`	2 Xeon 3 GHz	1 GB	1 Gbps	2.6.12.6
`turtle`	1 Ultra Sparc IIi	128 MB	100 Mbps	2.6.16.14

The SNMP command responder was running on `meat` and `turtle` for the measurements without packet loss and `veggie` was acting as a command generator. Under the condition of packet loss, the command responder was running on `turtle` while the command generator was running on `meat`. For the

SNMP/USM measurements, we used the authentication plus privacy security level with HMAC-MD5 as the authentication algorithm and AES-128 as the encryption algorithm. The `libssh` library was configured to also use the HMAC-MD5 authentication algorithm and the AES-128 algorithm for encryption and null compression.

We instrumented the `snmpget`, `snmpwalk` and `snmpbulkwalk` programs (part of the `Net-SNMP` package) to measure the latency by calling `gettimeofday()` before opening a session (but after parsing of MIB files) and after closing the session and computing the time difference. The output was directed to `/dev/null` and each experiment was repeated 100 times. The `tcpdump` tool was used to calculate the number of bytes exchanged and the `pmap` tool was used to measure the memory sizes of the processes. Packet loss was simulated by using the `netem` network emulation queuing discipline of the Linux kernel on `meat` and `turtle`.

The object identifier (OID) used for `snmpwalk` and `snmpbulkwalk` measurements was the interface table `ifTable` [8]. The object identifier used for `snmpget` measurements was the scalar `sysDescr` [9]. The number of rows in the `ifTable` was manipulated by creating virtual LAN (VLAN) interfaces.

5.2 Session Establishment

Table 2 shows the result of performing `snmpget` requests on the scalar `sysDescr` using different transports. There is a significant cost associated with the establishment of SSH sessions. This is, however, not surprising since the SSH protocol establishes a session key using a Diffie-Hellman key exchange (using public-key cryptography) before the user authentication protocol is executed and the SSH channel is established. Since the cryptographic operations are CPU bound, the session establishment times increases significantly on our slow test machine.

Table 2. Performance of `snmpget` requests (`sysDescr.0`)

Protocol	Time (`meat`)	Time (`turtle`)	Data	Packets
SNMPv2c/UDP	1.03 ms	0.70 ms	232 byte	2
SNMPv2c/TCP	1.13 ms	1.00 ms	824 byte	10
SNMPv3/USM/UDP	1.97 ms	2.28 ms	668 byte	4
SNMPv3/USM/TCP	2.03 ms	3.03 ms	1312 byte	12
SNMPv2c/SSH	16.17 ms	91.62 ms	4388 byte	32

Table 2 also shows that there is an clear difference in the amount of data (total size of the Ethernet frames) and the number of IP packets exchanged between UDP and TCP transports. While this overhead is usually not a big issue on a well functioning local area network, it might be an issue in networks with large delays or high packet loss rates.

5.3 Latency Without Packet Loss

Figure 3 shows the latency for the retrieval of the `ifTable` [8] with different sizes using `snmpwalk`. The difference between the transports UDP and TCP seems marginal compared to the difference caused by enabling authentication and privacy. The performance of the SSH transport is interesting. Initially, the costs of establishing an SSH channel with a new session key cause the SSH transport to perform worse than SNMPv3/USM. However, there is a break-even-point after ≈ 500 SNMP interactions where the SSH transport becomes more efficient than SNMPv3/USM. This observation seems to be independent of the speed of the machine hosting the command responder.

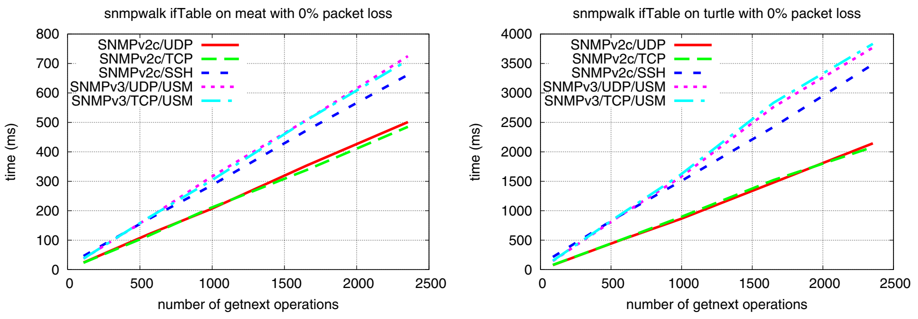


Fig. 3. Latency comparison of SNMP `getnext` walks (`ifTable`) without packet loss with a command responder on a fast machine (left plot) and on a slow machine (right plot)

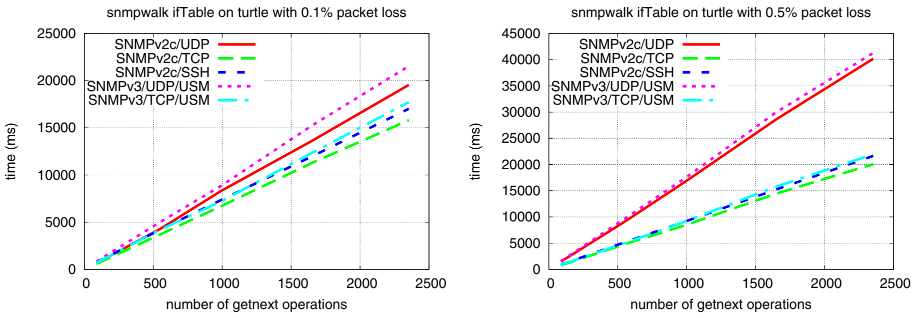


Fig. 4. Latency comparison of SNMP `getnext` walks (`ifTable`) under packet loss with a command responder on a slow machine with different packet loss probabilities

5.4 Latency with Packet Loss

Figure 4 shows the latency of the same `snmpwalks` on the `ifTable` with 0.1% and 0.5% packet loss using the slow command responder running on `turtle`. The surprising result is that the TCP-based transports all clearly outperform the UDP-based transports. It turns out that `Net-SNMP` has a very simple retransmission scheme with a default timeout of 1 second and 5 retries. TCP reacts much faster to lost segments in our experimental setup and this explains the bad performance of the UDP transports in the plots of Figure 4. Note that this result cannot be generalized since other SNMP implementations may have other retransmission schemes. However, simple statements that UDP transports outperform TCP transports in lossy networks are questionable as long as the application layer retransmission scheme is not spelled out.

5.5 Bandwidth

Figure 5 shows the amount of data exchanged during the `snmpwalk` experiments without packet loss. It can be seen that the amount of data exchanged increases when switching from UDP to TCP due to larger TCP headers. Furthermore, SNMPv2c/SSH requires less bandwidth than SNMPv3/USM/UDP and SNMPv3/USM/TCP. Even though the SSH connection protocol adds a tiny header, it seems that this header is significantly shorter than the space needed for the SNMPv3/USM header.

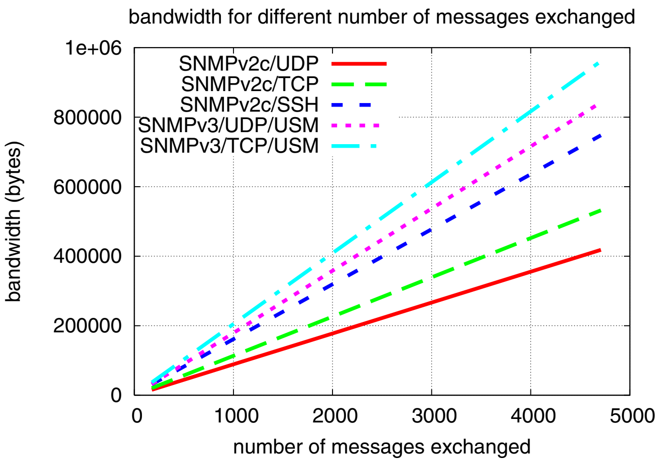


Fig. 5. Bandwidth used for different numbers of exchanged SNMP messages

Table 3 indicates that the change from SNMPv2c to SNMPv3/USM costs approximately 90 bytes of overhead per packet while the SNMPv2/SSH prototype adds approximately 40 bytes of overhead per packet to the SNMPv2c/TCP transport. While the SNMPv3 message header is slightly larger than the SNMPv2c header we have used in our implementation, we believe that also an SNMPv3/SSH implementation will consume less bandwidth than SNMPv3/USM when many packets are exchanged. As can be seen from the total number of packets, the piggy-backing of TCP ACKs worked nicely for all TCP transports.

Table 3. Packet sizes for the `snmpwalk` with 2354 `getnext` operations. The 'range' column shows the dominating packet size range and the 'packets' column the number of packets in that range; the column 'total packets' indicates the total number of packets.

Protocol	Range	Packets	Total Packets
SNMPv2c/UDP	80-100	4710	4710
SNMPv2c/TCP	110-130	4709	4712
SNMPv3/USM/UDP	170-190	4710	4712
SNMPv3/USM/TCP	200-220	4709	4714
SNMPv2c/SSH	150-170	4711	4742

5.6 Memory Usage

Figure 6 shows the amount of virtual memory used by the command responder process for an increasing number of concurrently open sessions. Initially, the process needed 9312 KByte of memory. The memory consumption grows linearly with the number of concurrently open sessions. The measured memory overhead per session is approximately 7 KByte. Note that the sessions were only established but not used during these measurements. Some additional memory might be dynamically allocated by the SSH library once data exchanges take place.

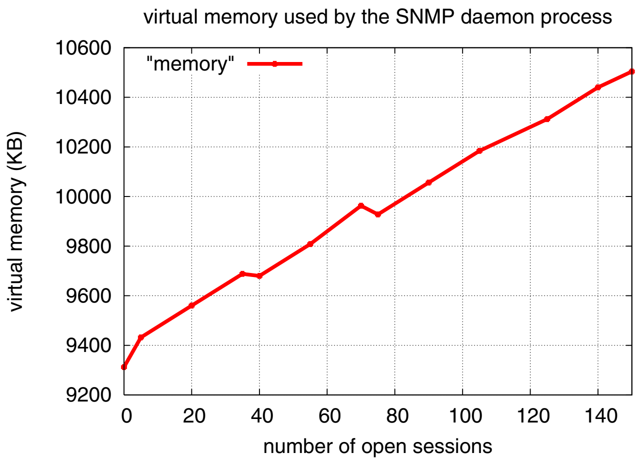


Fig. 6. Virtual memory consumed by the command responder process for an increasing number of open sessions

6 Related Work

SSH is already widely used to secure the access to command line interfaces on network elements. Accordingly, SSH credentials (keys, passwords) are readily available in many environments. This availability of credentials in many operational networks has been the main motivation for considering SSH as a secure transport for SNMP.

The obvious alternative to SSH is the widely used Transport Layer Security (TLS) protocol [10]. An implementation of SNMP over TLS has been analyzed in [11]. Although the authors used a very different setup, some key results are similar to the results reported in this paper. The authors of [11] also observed that SNMP over TLS is more efficient in terms of latency than SNMPv3/USM for longer sessions.

The short-coming of the work done on SNMP over TLS back in 2001 was that architectural questions were not considered and it thus remained unclear how model independent values such as the SNMP security name or security level are determined and passed around. These architectural questions have meanwhile been addressed in the IETF as described in [5] and summarized in Section 2.

The work reported in [12] compares the costs of using SNMPv3/USM with different security levels with SNMPv1 and SNMPv2c. While some of the measurements reported in this paper are comparable with results reported in [12] (e.g., the network capacity consumed for `snmpget`), the results are not identical and differ slightly. In particular, the overhead of SNMPv2/USM with authentication and privacy seems to be generally higher compared to SNMPv1/SNMPv2c in our measurements. This may be explained by the fact that the SNMP engines used in both studies are different as well as the operating system and hardware platform used. In addition, different MIB objects were fetched in both experiments: This study uses the `sysDescr` scalar, a `DisplayString` of ≈ 60 characters, while an 4 byte `IpAddress` object was used in [12].

7 Conclusions

The ISMS working group of the IETF is working on new security models for SNMP which leverage a secure transport. One crucial question is how the performance of this new approach compares to the existing security solution for SNMP (SNMPv3/USM) and to the still widely deployed insecure versions of SNMP (SNMPv1/SNMPv2c).

The measurements presented in this paper try to give answers to some of these questions. In particular, we quantified the session establishment overhead for SNMP over SSH. For simple one-shot SNMP requests, SSH seems to be a rather costly solution since the costs for establishing a session and associated session keys is significant. For sessions that carry multiple SNMP interactions (e.g., table walks), the costs for the initial session setup are amortized and there is a break-even-point where SNMP over SSH starts to become more efficient than SNMPv3/USM with authentication and privacy enabled.

The answer to the question whether SNMP over SSH is a viable alternative to SNMPv3/USM therefore depends on the SNMP usage pattern and the typical session length. While SNMP traditionally has no concept of a session, it is possible to approximate session life times by analyzing SNMP traffic traces. Work is underway in the Network Management Research Group (NMRG) of the Internet Research Task Force (IRTF) to collect SNMP traffic traces from different operational networks [13]. These traces are expected to give insights in which environments SNMP over SSH is likely to be a viable alternative.

Acknowledgments

The work reported in this paper was supported in part by the EC IST-EMANICS Network of Excellence (#26854).

References

1. J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and Applicability Statements for Internet Standard Management Framework. RFC 3410, December 2002.
2. U. Blumenthal and B. Wijnen. User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). RFC 3414, December 2002.
3. T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251, January 2006.
4. D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411, December 2002.
5. D. Harrington and J. Schönwälder. Transport Mapping Security Model (TMSM) Architectural Extension for the Simple Network Management Protocol (SNMP). Internet Draft (work in progress) <draft-ietf-isms-tmsm-03.txt>, June 2006.
6. D. Harrington and J. Salowey. Secure Shell Security Model for SNMP. Internet Draft (work in progress) <draft-ietf-isms-secshell-02.txt>, June 2006.
7. A. G. Morgan. The Linux-PAM Application Developers' Guide. Technical report, November 1999.
8. K. McCloghrie and F. Kastenholz. The Interfaces Group MIB. RFC 2863, June 2000.
9. R. Presuhn. Management Information Base (MIB) for the Simple Network Management Protocol (SNMP). RFC 3418, December 2002.
10. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, 2006.
11. X. Du, M. Shayman, and M. Rozenblit. Implementation and Performance Analysis of SNMP on a TLS/TCP Base. In *Proc. 7th IFIP/IEEE International Symposium on Integrated Network Management*, pages 453–466, Seattle, May 2001.
12. A. Corrente and L. Tura. Security Performance Analysis of SNMPv3 with Respect to SNMPv2c. In *Proc. 2004 IEEE/IFIP Network Operations and Management Symposium*, pages 729–742, Seoul, April 2004.
13. J. Schönwälder. SNMP Traffic Measurements. Internet Draft (work in progress) <draft-irtf-nmrg-snmp-measure-00.txt>, May 2006.

Uncertainty in Global Application Services with Load Sharing Policy

Mark Burgess and Sven Ingebrigt Ulland

Oslo University College, Norway
mark@iu.hio.no

Abstract. With many organizations now employing multiple data centres around the world to share global traffic load, it is important to understand the effects of geographical distribution on service quality. The Domain Name Service is an important component for global load balancing. Using controllable simulations, we show that wide area sharing can play an important role in optimization of response times when traffic levels exceed that which can be supplied by a local infrastructure. We compute the probability of being able to meet Service Level Objectives as a function of DNS caching policy (Time To Live), so that service providers can account for DNS error margins in Service Level Agreements.

1 Introduction

Meeting Service Level Objectives (SLO) is a crucial goal for online businesses, especially in the application services sector. Even if no formal Service Level Agreement (SLA) has been made, Service Level Objectives (SLO) are set by clients' demands: users will typically defect from a slow web-site after only a few seconds of waiting in order to look for an alternative[1], hence performance is directly related to profit.

One of the benefits that networking offers is the ability to use distributed resources to one's advantage. When local resources fail to cope with demand, external resources can be brought into play simply by redirecting requests to another location. That location could be a few centimetres away, or several thousand kilometres across a wide area network. However, in wide area, globalized services there are more links in the chain between server and customer where uncertainties and delays can creep in, and each of these components becomes a focus of independent interest for the performance analyst.

The role of the Domain Name Service (DNS) has been of particular interest in the matter of load sharing and redirection of traffic. On the one hand, this is an obvious place to overload existing functions for the purpose of load balancing; on the other hand it is a fragile bottleneck in Internet services with low security properties and relatively poor performance. Ten years ago, network service capacities were orders of magnitude poorer than today so the delays incurred by DNS lookups was less noticeable. Today, the DNS appears as a key bottleneck which contributes a significant fraction to service round-trip times.

In this paper, we examine the predictability of wide area load-sharing, based on the Domain Name Service, and attempt to identify strategies for minimizing its impact on Service Level Objectives.

2 Global Network Services

Global load sharing is a subtle topic, because it makes cost trade-offs based on quite different currencies. Many layers are involved in directing a service across the globe, and each of these can add to the uncertainty in response time, and to the delay experience by the end user. One must therefore try to unravel the various components.

Why would we not simply centralize a service for easy management? The reasons for this include security, fail-over redundancy, traffic congestion management or even power saving (or cost saving under different tariffs)[2]. The inhomogeneity of load that occurs during the course of a day often makes night-time processing favourable in a data centre[3], thus it might be a reasonable strategy to divert traffic to a geographically remote location (on the other side of the planet) in order to balance a heavy day-time load – this would assume that the routing and transport cost were acceptable[4,5]. In other work, we have looked at how local methods can be used to predict the scaling of application services in a data centre. The ability to respond to a request depends on both the nature of demand[6] and the availability of supply[7].

There is a need then to be able to predict the performance of wide area redirections for coping with server traffic, and study them in relation to the benefits of more local strategies. We approach the question of application service performance by asking a simple question: how does the use of wide area methods for network redirection affect application services levels? i.e. What level of uncertainty does globalized load balancing add to service response times?

There is a large literature on application service modelling. Much of it is now five to ten years old, and some is contradictory. We consider the situation today.

3 DNS

The Domain Name System (DNS) is the global name lookup service in the Internet. It is a distributed, hierarchical and redundant database running on many thousands of servers world-wide, each responsible for one or more DNS zones. When a client issues a request for a URL, the browser will first try to resolve the hostname in the URL into an IP address, so that it knows where to send the request. This is where it is possible for a DNS server to influence the outcome of a query, effectively directing the client to a desired or lightly loaded site. However, the DNS approach to load balancing is not without challenges, as will be discussed shortly.

DNS servers come in a variety of flavours all of which allow the service to act as multiplexers, serving different looked-up values for incoming requests in a one-to-many mapping. The approaches differ in their 'back-ends', i.e. the amount of internal processing they use to adapt to their situation. We shall consider both static and adaptive ‘back-ends’ in this paper.

Consider a single DNS query, and assume that its return value has not previously been cached anywhere. The following sequence of events ensues:

1. A client program passes a partially qualified hostname to a system interface, (normally called `gethostbyname()`.
2. The operating system *qualifies* the name by adding a local domain name, if none was specified, and redirects the request to the local resolver (typically a local DNS nameserver), asking for the A-record (or AAAA-record in IPv6) for the fully-qualified name – an A-record is a standard hostname-to-IP mapping. The request to the resolver is *recursive*, which enables a flag meaning “I only want the final answer.”
3. If the local resolver does not have the answer, it performs *iterative* queries through the DNS hierarchy. First it asks one of the thirteen root DNS servers. These know which servers control the top-level domains like com, org and net. The local resolver caches the response.
4. Further, the resolver asks one of the top level DNS servers for further directions to the domain. Again, it caches the response.
5. When the iteration process reaches one of the lookup domain’s DNS servers, this server will know the answer to the query, and reply with an IP address, e.g. 192.0.34.166. Yet again, the resolver caches the response.
6. The response is sent back to the client operating system, which also caches the response.
7. The IP address is returned to the browser, which in turn can contact the server and retrieve the content. In addition to the operating system caching the response, many clients will do so as well.

Note how this rather simple example introduces at least five levels of caching, not counting potential intermediate http proxies.

Consider a query to a balanced web site. A typical response could be as in this example:

```
;; QUESTION SECTION:
;cnn.com.                      IN      A

;; ANSWER SECTION:
cnn.com.              300      IN      A     64.236.29.120
cnn.com.              300      IN      A     64.236.16.20
cnn.com.              300      IN      A     64.236.16.52
cnn.com.              300      IN      A     64.236.16.84
cnn.com.              300      IN      A     64.236.16.116
cnn.com.              300      IN      A     64.236.24.12
cnn.com.              300      IN      A     64.236.24.20
cnn.com.              300      IN      A     64.236.24.28

;; AUTHORITY SECTION:
cnn.com.              600      IN      NS    twdns-04.ns.aol.com.
```

The nameservers return multiple A-records for the ‘cnn.com’ hostname – this list is known as a Resource Record Set (RR set). The addresses appear to be within the same provider network. This does not mean however that they are geographically close to each other. Clients typically traverse the RR set sequentially, starting from the top. That is, if the first address on the list does not work, the client tries the next one, and so on. This is a decision that the client makes.

The numbers in the second column of the response show the Time-To-Live integer value (TTL) in seconds. This value governs how long the answer can reside in a cache in any of the intermediate nodes. Once this value has expired, the cache discards the value and the client must obtain an ‘authoritative’ answer by iterative querying once again. This is a traditional strategy for off-loading DNS servers and reduce lookup latency by caching the IP address value of a DNS lookup for the specified lifetime. The relevance of this mechanism must be reevaluated in light of performance improvements over the intervening decades.

As long as the TTL is greater than zero, queries will be answered from a cache instead of being redirected to other servers. In the example, the A-records have a TTL of 5 minutes. A low TTL ensures that DNS servers are queried often, and hence are given the possibility to obtain fresh, up-to-date information about a domain which is making changes over relatively small time intervals. Low TTLs mean frequent repetition of the arduous look-up process and hence come at the cost of adding a considerable delay to the total service response time.

For example, the DNS can consume a significant part of the time it takes to fetch a web page, which might have several in-lined objects, including pictures and advertisements from many source domains. Shaikh et al note: “25% of the name lookups (with no caching) add an overhead of more than 3 seconds for the ISP proxy log sites, and more than 650 ms for the popular sites. respectively. It is interesting to observe that nearly 15% of the popular sites required more than 5 seconds to contact the authoritative nameserver and resolve the name. This is likely to be related to the 5-second default request timeout in BIND-based resolvers” [9].

As we shall see, caching alleviates this problem considerably (indeed, the question of caching versus non-caching leads to a strongly bimodal behaviour), but with a different potential cost: global congestion.

From the above, we note that DNS can employ two basic mechanisms in multiplexing: it can cyclicly permute the RR set so that each new query is ordered differently. Since clients normally pick the first element from the list, this amounts to a continual Round-Robin shuffling of the server set. Second, the TTL value must be chosen to be a relatively low value to avoid too much re-use of old data which would bias the fair-weighted Round-Robin distribution.

Clearly, these methods are unreliable. There is much uncertainty. If we have three servers in the RR set and only every third query from a given client contains a service request, then all the traffic goes to the same server after all. We are also trusting clients to act predictably and not try to second guess the results from the DNS server by performing their own shuffling: DNS implementations are not required to preserve the order of resource record sets. As for TTL values, multiple levels of caching make it a challenge to predict and control the actual

TTL observed by the end user. The TTL policy is only a polite request to caches, not an enforcable mechanism.

A problem with low TTL values is in so-called *sticky sessions* in which a user is connected to a particular instance of a network service with a cookie of session identifier that is unique to one server (e.g. in a net bank or online retailer). If the next request to a persistent session were directed to a different server, the session would be lost. We shall not discuss this particular issue here, since its resolution is a story in its own right.

4 DNS Latency

Service Level Objectives are performance wishes often set informally by clients and estimated by engineers. The service provider wants to guarantee that users will gain access to their services within a Service Level Objective. Users are known to give up on slow services within just a few seconds and take their custom elsewhere[1].

The uncertainty is the response time is rightfully a combination of the uncertainty margins in each of the independent causal factors between the client and the server. This includes the identification of the appropriate server through the DNS service. One would like to write:

$$\Delta t = \sqrt{(\Delta t_{\mathrm{DNS}})^2 + (\Delta t_{\mathrm{Routing}})^2 + ...} \tag{1}$$

Since the DNS is a network service, it is itself dependent on all of the other uncertainties in a network, so each DNS query invokes all of the latencies in the system even before a service has commenced. Alas, the inter-dependencies of a network make the Pythagorean formula above difficult to implement.

A data centre engineer would like to attempt to compensate for the lookup delay, or at least account for it in service promises to clients. Let us consider then, how much DNS server redirection could add to the margins needed for the 'over-provision' of services, according to this study? On a global scale there are more sources of possible delay which could add to the overall response time. It is plausible that the worst bottleneck would be shifted to some other component in the supply chain. Hence it seems possible that one might win performance improvements by making a more informed choice based upon monitoring of the available capacities along different alternative paths.

Consider the scenario in figure 1. We imagine a global organization with alternative data centres at different locations. Redirection to the appropriate data centre will occur by DNS multiplexing. There are two competing mechanisms in such a load sharing scheme: the desire to avoid bottlenecks at the dispatcher and the desire to avoid congestion at the servers themselves. If the dispatcher does not share efficiently, there might be congestion, but if the dispatcher struggles to share the load it could add to the service time itself. This is a classic problem in inventory management[10].

The DNS Time To Live is key here, since a high TTL lowers the load on the DNS as a dispatcher, but at the same time increases the congestion on the

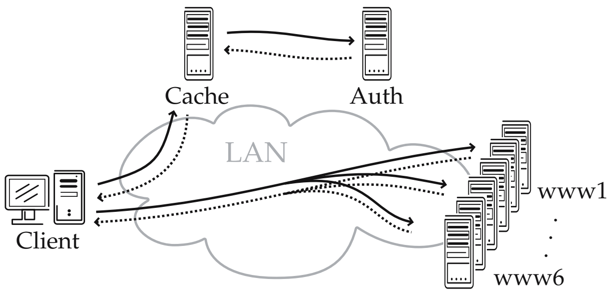


Fig. 1. A schematic illustration of the DNS load sharing scenario

servers. It behaves as a slider-knob trading off these two effects. Figure 2 shows what one might expect for the relationship between the TTL value set in the authoritative nameserver and the round-trip time of fetching web-objects from a hostname that is registered with several IPs in the authoritative nameserver.

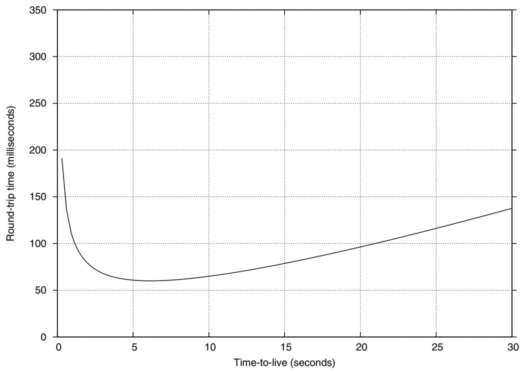


Fig. 2. Hypothesized cost of a DNS based load balancer at high traffic levels. The cost rises for low TTL as the load on the DNS server increases. The cost rises for long TTL as the individual servers start to become loaded inefficiently due to unfair weighting.

What is interesting about this form is the possibility that there exists an optimal value for the TTL parameter (the inventory re-order time). In previous work this TTL parameter has been set essentially by hand, without much insight into its functional role[8,11]. We would like to investigate this causal role of the TTL value more carefully below by testing DNS implementations against simulated traffic patterns.

5 Empirical Study

Our experiments investigate the policies that can minimize DNS induced latency. Such latency can come from the inefficiency of the DNS service itself, from the relative congestion of alternative servers, and from transport uncertainties (which are unrelated to the DNS). We arrange for the latter to disappear by isolating a DNS load balancing scenario in the lab. Hence we are left with the interplay between dispatcher (the DNS response time) and server congestion (determined by the algorithm used by the dispatcher) which yields a final service time.

We deploy a client, the client's local resolver with cache (representing a local domain nameserver), and the remote domain's authoritative DNS server. We also have six web-servers, all configured similarly to answer requests for a single hostname, e.g. `www.example.net`. Wide-area network emulation is provided by the NetEm facility in the 2.6.16 linux kernel[13]. This is a part of the QoS framework there, and it allows us to specify delays and delay distributions for outgoing queues to simulate load. In the experiments on TTL vs RTT, the cache has a mean delay of 20ms, the authoritative has 300ms.

5.1 Effect of TTL on Round-Trip Time, Homogeneous Servers

Running the `flood` tool to test DNS server response allows us to test our hypothetical inventory processing model for the combined lookup and service time. For the initial test, we make all of the servers identical in capacity and latency.

The plot in figure 3 shows the effect of TTL on static DNS server response-time for a full page load, that is, a DNS request for the hostname, TCP connection setup, sending HTTP request and finally receiving HTTP reply (connection teardown is not included). The HTTP request used is trivial to serve and requires few CPU cycles per request.

For TTL 0, there is little variance in the results. We found that it was essentially impossible to overload a DNS server running on modern hardware with realistic traffic intensities. The uncertainty bars corresponds exactly to a controlled delay distribution which we specified for the authoritative nameserver (using NetEm).

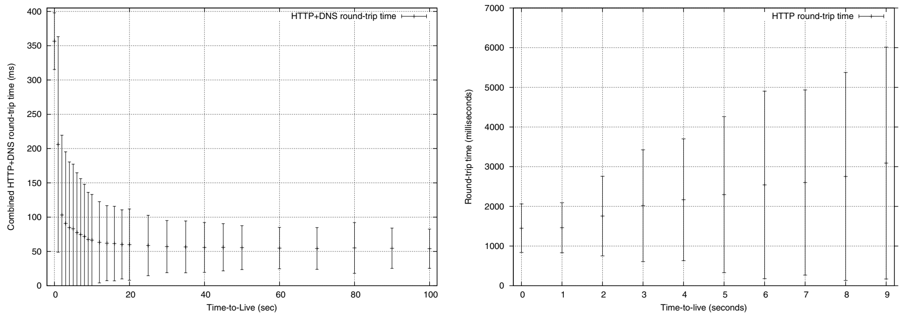


Fig. 3. Cost of lookup as a function of TTL for a 'static backend' DNS server. To be compared with the hypothesized form. Rather than rising at the end, it flattens out. The second graph has no low TTL lookup cost up shows a rising inefficiency cost as server balancing fails.

For TTL 1 and beyond, DNS requests are served both by the authoritative and caching nameserver. Since these two servers have very different delays set up for their outward queues, a request would either be served by the cache or the authoritative server. As TTL increases, the chance for a cache hit in the caching server increases proportionally. We see a flattening out of the tail for large TTL and no apparent rise in server congestion, since the load presented by our test page was low. Thus, this data represents primarily the behaviour of the dispatcher.

Figure 4 shows how the DNS response time distribution is strongly bi-modal, clearly showing the influence of caching. The figure is dislocated so we can view the two peaks in more detail. We see that for TTL 0, there are no cache-hits on the caching server. But as TTL increases, so does the cache-hit rate. For TTL 100, we see a very high number of cached replies, and a close to zero amount of time-intensive authoritative requests.

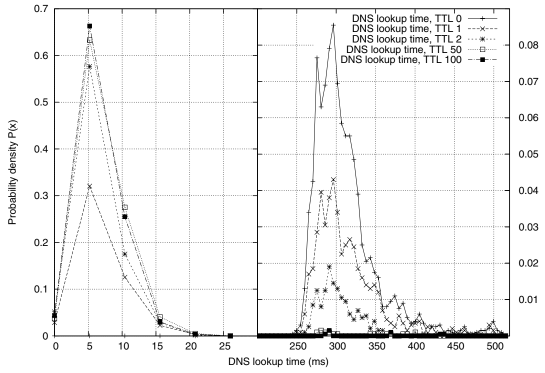


Fig. 4. The bimodal response time distribution for DNS showing the effect of caching

The above experiment was repeated for a dynamic back-end DNS server: PowerDNS (with PostgreSQL as back-end). Here the database is loaded with the zone-data containing the resource records. PowerDNS is set up to query the database using a simple query, and appending `"ORDER BY random() LIMIT 1"`, which returns one random IP address when requesting the hostname. The linearly increasing uncertainty in second figure 3 can be attributed to server loading due to poor entropy. Requests are queued and the system enters a thrashing phase. It increases with the TTL since a low TTL spreads the requests very efficiently among the servers, avoiding overload. As TTL increases, the probability of server load does too, and thus also the uncertainty. Thus we have measured both tails of our hypothetical curve for different traffic regimes.

5.2 Distribution Entropy

The plot in figure 5 shows the cumulative frequency distribution of overall response times. What we observe is that for a zero TTL, most requests (90%) lie within the range of 0 to 1000 milliseconds per request. For TTL 9, we can see that approx 60% of the requests lie within the range of 0 to 2000 milliseconds

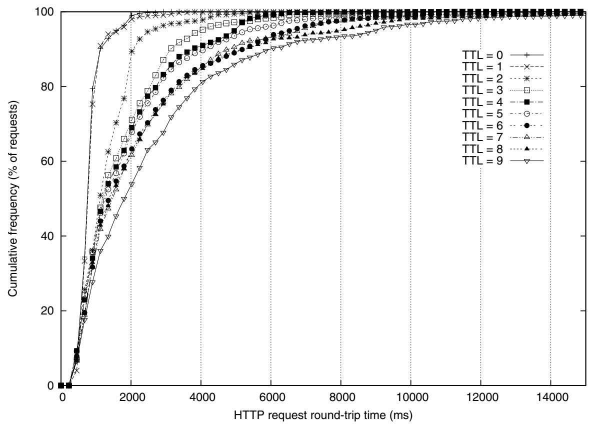


Fig. 5. Cumulative frequency plot of service times show with what certainty we can specify a response time as a function of TTL

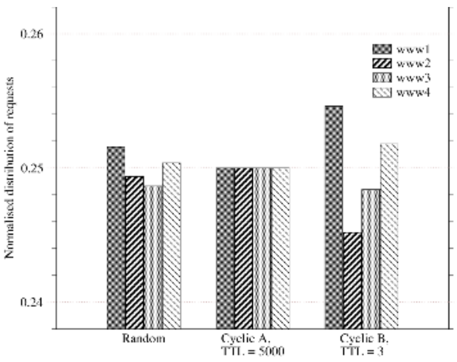


Fig. 6. Entropy of load balancing by DNS for one client

per request. Also for TTL 9, we see that approx 90% of the requests lie within the range of 0 to 6000 milliseconds per request. Note that a long TTL seems to imply a longer wait, which warns against the very large TTL values suggested by authors several years ago. These results are of great interest to the Service Level Agreement architects. The efficiency of the load scheduler itself also depends on the TTL. Ideally a load balancer will maximize the entropy of the connections histograms[14]. Figure 6 shows how well BIND distributes requests evenly among servers in a resource record set for a single client.

In the"Cyclic B" case, with a TTL of three seconds we see a more uneven response caused by BIND resetting the order of the resource record set each time the TTL expires. This causes a greater uncertainty which the simple round-robin algorithm does not cope well with: an unfair weighting is induced on the server record distribution.

The variations are very small, however, and do not pose any risk of over-utilisation for server 'www1' unless there is critically high traffic, in which case a non-linear instability could be seeded by this lack of parity. However, seen in the context of requests arriving from a source of many clients in diverse domains, there could be sufficient entropy of clients to even out this behaviour.

6 Comparable Work

Several researchers have examined load balancing using DNS with varying conclusions. Cardellini et al survey proposed and commercially available balancing schemes, including constant and adaptive TTL schemes for DNS, dispatcher-based packet rewriting, and server-based mechanisms[8]. They find that both constant TTL with server and client state information, and the adaptive TTL scheme perform better than stateless round-robin approach.

Bryhni et al also compare a set of load-balancing implementations, with a focus on dispatcher-based systems[11]. The round-robin DNS is discussed with TTLs of 1 and 24 hours. Their trace-driven simulation results show that a dispatcher-based design running a round-robin algorithm yields the best distribution of load and amongst the lowest response times observed for that particular scenario.

In another paper Cardellini et al present a more thorough examination of web-server load balancing using DNS, and introduce HTTP redirection as a potential remedy for the otherwise coarse-grained nature of DNS[15]. They claim superior performance of redirection mechanisms over classic DNS-only balancing.

Shaikh et al, show that lowered TTL values must be carefully chosen to balance page responsiveness against excessive latency observed by the client[8]. The authors recognise that, to allow a fine-grained and responsive DNS-based server selection scheme, the TTL should be set to zero or a very low value, however this can lead to two orders of magnitude of extra delay, according to the paper. Other authors also explore the effectiveness of lowered TTL values. Jung et al [16], Teo [17], Park [18].

7 Discussion and Conclusions

DNS load balancing is a somewhat controversial topic. We have examined the behaviour of the DNS implementations with regard to their caching policy in order to find the expected uncertainty in meeting Service Level Objectives.

We find that today's DNS servers easily cope with high request volumes, in high levels (notwithstanding denial of service attacks). Caching policy does not impact directly on performance from the viewpoint of the server. However, the response time of the DNS service is relatively high by comparison to other services, due to the iterative nature of queries.

Round robin load balancing in DNS service works adequately with high levels of entropy, but are more likely to become unstable under high traffic conditions for low TTL. Low level load balancers favour simple round-robin load-sharing at low to medium intensity[7]; there one has very low latency routes between the dispatcher and server and the cost of looking for improvements outweighs any benefits. In global routes, a DNS server can benefit from a knowledge of the round-trip time when load balancing. Unfortunately, there is not a clear correlation between the time measured by the client, and that measured by the DNS server load balancer, so this does not work well.

The uncertainties inherent in wide area load sharing mean that a DNS load balancing strategy is not a substitute for low level load-sharing mechanisms. Failover redundancy is a main reason for having multiple data centres, but this is not the same as load balancing by DNS. Cumulative frequency plots indicates the additional round-trip time with corresponding uncertainties for requests. This shows us what one can expect to achieve in an agreement 80% or 90% of the time. Pre-sorting sites, e.g. by 'picking the site in your country' etc preempts DNS weaknesses.

DNS cannot be avoided, but is it the right tool for load balancing? Clearly it is not. However, it is nearly the only viable *interface* for load-balancing on a global scale (IPv4 anycast is another). DNS, with the solid backing of a dynamical back-end (based on a database with state-information gathered from the servers, maybe proximity information from ARIN's IP-to-country mappings, time zone info, etc), is a very powerful tool for global server load balancing.

We are grateful to Jon Henrik Bjørnstad and Gard Undheim for helpful discussions. This work is supported by the EC IST-EMANICS Network of Excellence (#26854).

References

1. J. Sauve et al. Sla design from a business perspective. In IFIP/IEEE 16th international workshop on distributed systems operations and management (DSOM), in LNCS 3775.
2. M. Burgess and F. Sandnes. A promise theory approach to collaborative power reduction in a pervasive computing environment. In Springer Lecture Notes in Computer Science, page to appear.
3. M. Burgess, H. Haugerud, T. Reitan, and S. Straumsnes. Measuring host normality. ACM Transactions on Computing Systems, 20:125160, 2001.
4. B. Abrahao et al. Self-adaptive sla-driven capacity management for internet service. In Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006), pages 557568. IEEE Press, 2006.
5. M. Burgess. Probabilistic anomaly detection in distributed computer networks. Science of Computer Programming, 60(1):126, 2006.
6. J.H. Bjrnstad and M. Burgess. On the reliability of service level estimators in the data centre. In Proc. 17th IFIP/IEEE Distributed Systems: Operations and Management (DSOM 2006), volume submitted. Springer, 2006.
7. M. Burgess and G. Undheim. Predictable scaling behaviour in the data centre with multiple application servers. In Proc. 17th IFIP/IEEE Distributed Systems: Operations and Management (DSOM 2006), volume submitted. Springer, 2006.
8. V Cardellini and M Colajanni. Dynamic load balancing on web-server systems. Internet Computing IEEE, 3(4):2839, 1999.

9. Anees Shaikh, Renu Tewari, and Mukesh Agrawal. On the effectiveness of dns-based server selection. In Proc. of IEEE INFOCOM 2001, Anchorage, AK 2001.
10. H.L. Lee and S. Nahmias. Logistics of Production and Inventory, volume 4 of Handbooks in Operations Research and Management Science, chapter Single Product, Single Location Models. Elsevier, 1993.
11. E Klovning H Bryhni and O Kure. A comparison of load balancing techniques for scalable web servers. 14(4):5864, 2000.
12. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In SOSP 03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 164177, New York, NY, USA, 2003. ACM Press.
13. Stephen Hemminger. Network emulation with netem. In LCA national Linux conference 05, 2005.
14. M. Burgess. Analytical Network and System Administration - Managing Human-Computer Systems. J. Wiley & Sons, Chichester, 2004.
15. Valeria Cardellini, Michele Colajanni, and Philip S. Yu. Geographic load balancing for scalable distributed web systems. In MASCOTS, pages 2027, 2000.
16. Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. Dns performance and the effectiveness of caching. IEEE/ACM Trans. Netw., 10(5):589603, 2002.
17. YM Teo and R Ayani. Comparison of load balancing strategies on cluster-based web servers. Transactions of the Society for Modeling and Simulation, 2001.
18. Kihong Park, Gitae Kim, and Mark Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In ICNP 96: Proceedings of the 1996 International Conference on Network Protocols (ICNP 96), page 171, Washington, DC, USA, 1996. IEEE Computer Society.

Predictable Scaling Behaviour in the Data Centre with Multiple Application Servers

Mark Burgess and Gard Undheim

Oslo University College, Norway
mark@iu.hio.no

Abstract. Load sharing in the data centre is an essential strategy for meeting service levels in high volume and high availability services. We investigate the accuracy with which simple, classical queueing models can predict the scaling behaviour of server capacity in an environment of both homogeneous and inhomogeneous hardware, using known traffic patterns as input. We measure the performance of three commonly used load sharing algorithms and show that the simple queueing models underestimate performance needs significantly at high load. Load sharing based on real-time network monitoring performs worst on average. The work has implications for the accuracy of Quality of Service estimates.

1 Introduction

Network services are now a central paradigm in the management of commercial resources in a market framework. They are divided broadly into network level services (such as sale of subscriber lines) and application level services (such as Internet banking and other Web-based portals). Predicting the expected service delivery is an important prerequisite for selling services in a market place; these expectations are then codified in Service Level Agreements in order to place the relationships on a contractual footing[1]. In simplistic terms, a service provider's task is to deliver the promised service level in as lucrative a way as possible (the Service Level Objective). There is a clear need for models which can reliably predict the relationship between supply and demand.

Meeting the requirements set by SLAs, with variable demand and resources, has become a keen point of interest in the last year[2,3,4]. A common strategy for trying to determine this relationship is to opt for "over-provision", i.e. providing more resources than are needed, by some acceptable margin. This is a straightforward way of dealing with uncertainty, but it can be a relatively expensive way and, without a model for capacity, it is not clear exactly what margin is required for meeting the peaks in demand. Another approach would be to try to adapt the service scheduling strategy and resource pool to better adapt to changing requirements[3,4]. In each case, there is a role for parallelism in the provision of the services both for the purpose of *load sharing* and *redundancy*.

In this paper, we consider how well one is able to predict the scalability of an application service by means of low level load balancing. We examine how the simple queueing models behave as estimators of local load balancing techniques, and we look at how response times vary as a function of traffic and number of servers. This information is of direct interest to data centre managers, especially when the hardware in a data centre in inhomogeneous, with varying processing capacity.

We quantify the expected uncertainty in service levels for different kinds of demand and equipment, so as to provide an estimate of service level margins, suitable for inclusion in a Service Level Agreement.

2 Experiment

Consider the following problem. Suppose our data centre resources are not fixed, but that we can add additional computing nodes in a cluster-like fashion, how can we adapt to changing service levels using models to predict the effects of changing the number of servers used in parallel?

We base our experiment on simulated traffic running on real hardware. The traffic is a stream of page-lookups to PHP-enabled web pages, generated using `httperf`. Each page involved CPU-bound iteration (spin delay); this was necessary to offer the server any load at all, as the server CPU hardware was comparable to that of the traffic generator.

There is some controversy in the literature about the exact arrival characteristics of Web traffic requests and how this affects the ability of a server to process them. We do not wish to involve ourselves in that question here (see [5] for a treatment of this matter); rather we wish to answer a far less ambitious question: assuming that the traffic follows the simplest, most predictable pattern, are we able use the corresponding models for load sharing to predict the performance of the balanced system? As we shall see below, this is far from being a straightforward question.

In our tests, we use a commercial Alteon 2208 load balancing switch as the dispatcher, to a back end of symmetrical IBM blade servers running Apache2 on GNU/Linux. These servers can be made identical, and they can be switched into power saving modes of lower CPU frequency (from 1.2 GHz to 2.8 GHz, stepwise). We make use of this feature to simulate the transition from homogeneous to inhomogeneous server hardware.

We would like to be able to predict the performance of servers in a data centre, with regard to

- Average delivery time of a job in the system.
- Change in average delivery time on adding new hardware.

This information is of direct use to a data centre manager adapting to demand, and it is also useful to the sales staff who decide when to promise specific service levels in their agreements.

Since we are dealing with random processes, exact prediction is not an option. A realistic picture is to end up with a probability or confidence measure e.g. what is the likelihood of being able to be within 80% or 90% of an SLA target value?

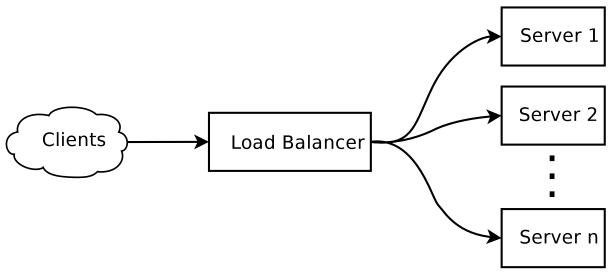


Fig. 1. The topology for low level load sharing using a commercial dispatcher

The shape of this probability distribution will also answer the question: what is the correct level at which to 'over-provision' a service in order to meet the SLA requirements. Fig. 1 shows the schematic arrangement of servers.

3 Queues and Service Prediction

Queueing theory is the domain of experts, not of data centre engineers. Indeed, in engineering terms, only the simplest queueing models are really used as predictors[6], because the more complex models require input that is not available to the average engineer. What is the distribution of traffic at a given moment? Does it vary throughout the course of a day or a week? These are questions that are not easily answered and, even if they are, those answers are not easily used in an advanced queueing model to give a simple result. We need therefore simple models that give approximate rules of thumb.

Recent work by Sauvé et al discusses the matter of trying to convert low level knowledge of a system into SLA margins, in a multi layered model using the cost of the equipment and services to add an element of risk analysis to the problem[2]. They use the simplest kind of queue model to estimate capacity – this is understandable given the complexity of the problem. However, they do not evaluate whether the model actually has any predictive power. Another group has considered the idea of combining queueing models with an adaptive control framework to allow service capacity to change with demands[3]. This work is rather interesting from a design point of view, but it does not answer the basic questions that an engineer might ask, such as how to deal with the varying demand in lieu of this new technology.

The basic queueing models we consider are, in Kendall notation, the $M/M/1$ queue and the $M/M/n$ queue for n servers[7]. The M stands for 'memoryless', i.e. discrete time Poisson inter-arrival time traffic. This is the 'simplest' case for queues, since Poisson traffic has simple analytical properties. It is known that Web traffic is rarely this well-behaved in practice, however, we use it as a source with the rationale that, if we cannot predict Poisson traffic, then we certainly cannot deal with more long-tailed traffic patterns. There is evidence to suggest that it is good enough to compensate for this model with additional uncertainty[5].

The total quality of service in a system must be viewed as the combined qualities of the component parts[8]. It is a known result of reliability theory[9] that low level parallelism is, in general, more efficient than high level parallelism, in a component-based system. Thus a single $M/M/n$ queue is generally superior in performance to n separate $M/M/1$ queues (denoted $(M/M/1)^n$). The $M/M/n$ queueing model is already significantly more complex than the $M/M/1$ model. A natural question is then: how to actual measurements of performance tally with either of these models, given a Poisson traffic source?

4 Server Performance

For calibration, we begin by considering the response of a single server to increasing load, as a control. The graph in fig. 2 shows a rising CPU utilization on the left of the graph, and a response time which stays low until we reach approximately 100 requests/second.

Notice that the response time eventually levels out, as packets are simply dropped above the threshold. Above 110 requests/second the response time stays between 1200 and 1400 ms. From these results it is clear that the server can handle somewhere between 100 and 110 requests per second, and that this task is heavily CPU bound (this is a result of our experiment design, but we believe that it is representative for many application tiered web services).

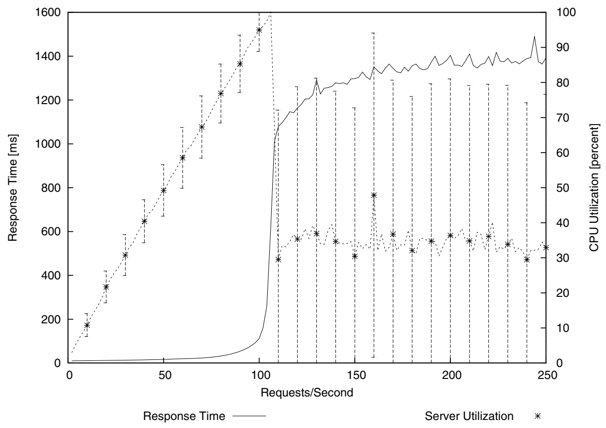


Fig. 2. Response time and CPU utilization of a single server as a function of requests per second, using Poisson distribution with exponential inter-arrival times

We also see that the CPU utilization increases linearly until around 105 requests/second, then it drops. This is a telling result since the server should be heavily utilized above this rate if the system is working efficiently. Looking at the standard deviation of the results obtained for CPU load, we see that it is very high when the request rate is above 110 requests/second. This implies that the server is unable to deliver the maximum level of service when requests per second exceed the maximum threshold due to thrashing at the server side. This is a strong indication that we want to avoid this region at all costs.

The response time also flattens out when the load exceeds approximately 105 requests per second. Above this rate the server starts to refuse connections instead of accepting all of them. Data show that the error rate above 100 requests per second increases proportionally with the increasing request rate. This again indicates that the server can't process request at any higher rate.

The reason for the results shown in fig. 2 is the queue length configured on the Apache web-server. Apache only spawns a limited number of child processes to handle incoming requests, and when all these processes are busy it needs to queue up waiting requests. This queue has a maximum length, in our case defined by the Apache `ListenBacklog` directive. The errors start to increase when the server has filled up its queue; 120 requests/second gives an error rate of 5.91 packets, and when the request rate increases above this we see that the increase in errors is almost the same as the increase in requests per second.

5 Load Sharing

Our load balancing arrangement uses a dispatcher at the bottleneck to distribute load to the back-end servers, using one of three different sharing algorithms:

- Round Robin: the classic load sharing method of taking each server in turn, without consideration of their current queue length or latency.
- Least Connections: the dispatcher maintains state over which back end server currently has fewest on-going TCP connections and channels new arrivals to the least connected host.
- Response Time: the dispatcher measures the response time of each server in the back end by regularly testing the time it takes to establish a connection. This has many tunable parameters.

In elementary queueing theory, there is no simple way of dealing with the issue of inhomogeneous servers. A simple question is therefore whether the simplest classical queueing models provide sensible predictive power for the data centre engineer in either a homogeneous or non-homogeneous case, on average.

6 Queueing Models as Sharing Predictors

We consider how queueing models perform compared to results obtained from our lab experiment. We use a c script to calculate response times using both $M/M/1^n$ and $M/M/n$ queues. From the previous sections we have calculated the performance of a single web-server, and we use this result in our queueing simulation. Below is a list of parameters used for expected response times using queueing theory:

- μ This is the variable we calculated when doing performance tests against one server. It is the processing capability of the server, also called the *service rate* and is often expressed in completions per millisecond. We take $\mu = 0.10413$from the fact that each server is able to process 0.10413 requests per millisecond.
- λ This is the *arrival rate*, and is expressed in arrivals per millisecond
- **n** Number of servers available. In our experiments we had at most 5 available servers to balance the load between.
  The *max_rate* specifies the limit at which we stopped simulation. This is usually equal to $n{\cdot}\mu$ since the queueing algorithms do not produce well-defined results when the servers are overloaded. In figure 3 max_rate equals $2 \cdot \mu = 0.20826$.

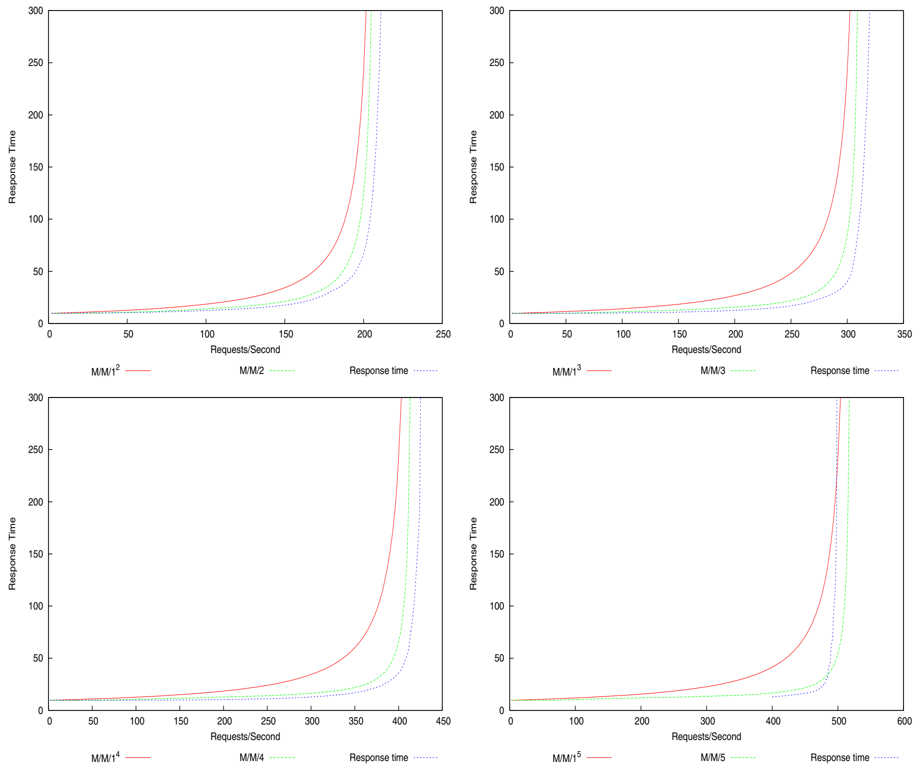


Fig. 3. Theory vs. Experiment: This graph shows the results obtained from simulating load balancing with both $M/M/n$ and $(M/M/1)^n$ queues and response times from a real experiment

Figure 3 shows a comparison of queue model with experiment for 2-5 indentical servers, using the Round Robin algorithm to balance the load between two servers. We see that the experimental results for n servers follow those for the $M/M/n$ queue quite closely up to the point of about 90% saturation. This is an encouraging sign for theory – given that the servers truly are measurably identical. It is perhaps better than one would expect given the simplicity of the queueing model. However, there is an indication that the model worsens as the number of servers increases. The $(M/M/1)^n$ is overly pessimistic for all n at low traffic, but $M/M/n$ traces the actual behaviour with surprising accuracy.

At about 90% of the maximum expected capacity (as calibrated against the single server levels) the models begin to fail. A summary of requests per second, errors and network input/output in Table 1 reveals that errors first start to occur when the request rate exceeds this 88% mark. The actual 2,3,4-server solutions even out-perform the $M/M/2, 3, 4$ predictions across the whole spectrum of loads, even when entering the thrashing regime. This is likely due to the fact that each server in fact deals with requests using parallel processing threads, not a First Come First Served (FCFS) discipline as the queue model assumes[5]; this is more efficient than FCFS. Nonetheless, the closeness of behaviour is notable for two servers. As the total load rises in the 5 server case, this behaviour changes and the thrashing regime cuts in even more sharply. This could be a sign of a change in the performance of the load balancer itself.

Table 1. A summary of CPU utilization, request and error rates when using Poisson distributed inter-arrival times in a 2-server regime

No. of requests	Response Time	Std. Dev.	No. of errors
2	9.89	0.09	0
10	9.87	0.12	0
50	10.45	0.13	0
100	12.49	0.12	0
150	17.64	0.36	0
160	20.12	0.67	0
170	24.40	1.00	0
180	31.40	1.72	0
190	41.08	3.27	0
200	66.04	18.35	0
202	78.44	34.21	0.002
204	97.91	77.31	0.003
220	853.56	76.73	0.1

Repeating the simulation for higher numbers of shows a similar story, but one which becomes progressively worse earlier. Figure 3 shows the comparison for $M/M/1^5$ and $M/M/5$. Here performance reaches approximately 490 out of the expected 520 requests per second (77%) before significant deviation sets in. Then at about 490 requests/second the experimental system reaches an instability and diverges drastically from the theoretical models. The fact that this occurs more quickly here is an anomaly; alas, we did not have more identical servers to attempt to extend the number to an even higher level, but this would have been interesting. Clearly, when theory and practice do not agree, some of the theoretical assumptions are incorrect. It is unlikely that the problem lies with the distributions. A clue as to the reason is found in the next section.

7 Scalability with Increasing Servers

If performance scales linearly when adding servers it means that we can easily predict the effect of adding more servers. This would be a useful property when considering QoS and maintaining SLA agreements. In figure 4 we show the addition of servers at four traffic levels intended to saturate the same number of servers.

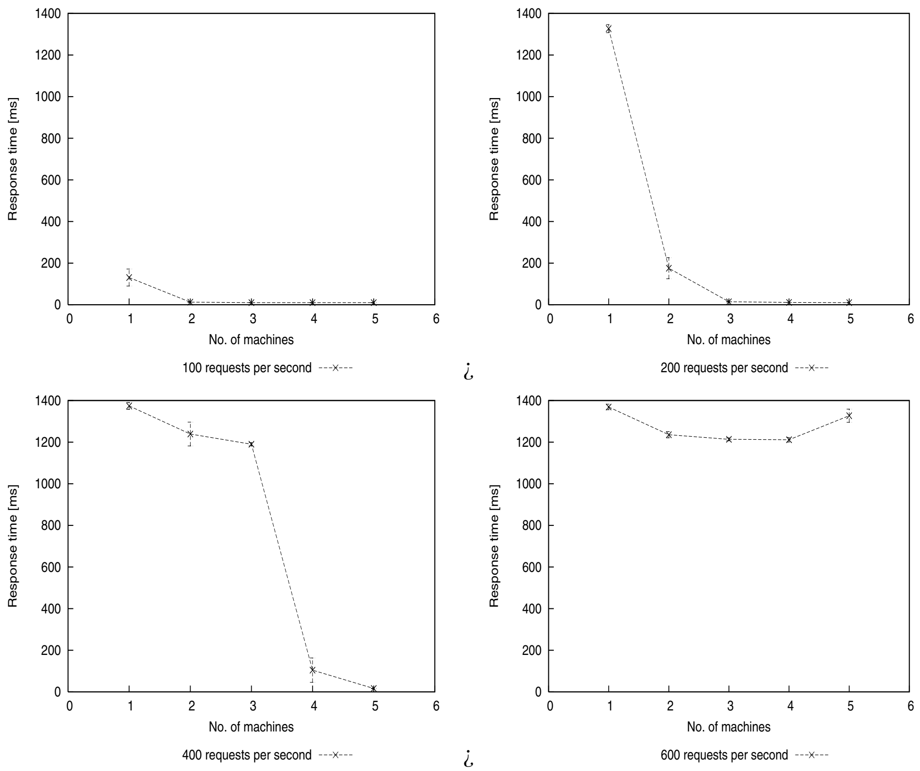


Fig. 4. How response rates scale when adding extra servers. For low traffic, adding a new server results in a discontinuous improvement in response time. At very high load, adding a server seems to reduce performance, leading to the upward curve in the last figure as the limitations of the bottleneck dispatcher become apparent.

In the first graph we see that the response time is kept low in all scenarios. This fits with the assumption that even a single server should be capable of handling 104 requests per second. The next graph shows that all scenarios except the one with a single server are capable of handling 200 requests per second. This fits the scaling predictions. With 400 requests per second, in the third graph, four servers just cut it. In the fourth graph we request 600 connection per second, and we see that the response rate is high for all scenarios. These results make approximate

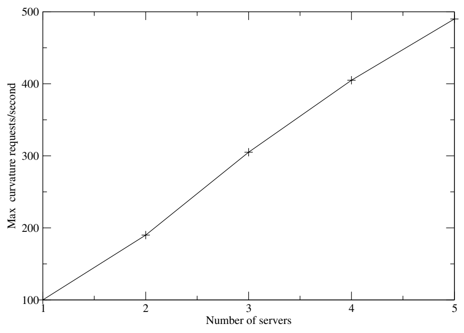


Fig. 5. Maximum processing capacity as a function of number of servers

sense, yet the scaling is not exactly linear; there is some overhead when going from having 1 server to start load balancing between 2 or more servers. We explore this further below. Figure 5 shows how the rates scale with number of servers. Note the discontinuities in points of maximum curvative in the response functions after one server and four servers. the response is not exactly linear.

8 Load Sharing Performance

The measurements above have been made with the round robin sharing strategy. It makes sense that this would agree maximally well with the $M/M/n$ model when all of the servers are indentical. In any real data centre, the likelihood of

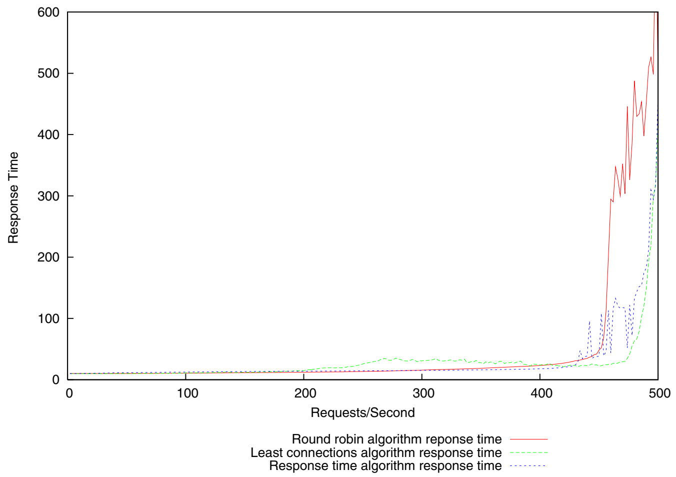


Fig. 6. The performance of the RT, LC and RR algorithms in a homogenous server environment. RR is superior at low traffic intensity, LC survives to slightly higher levels, but a small instability around 250 req/s.

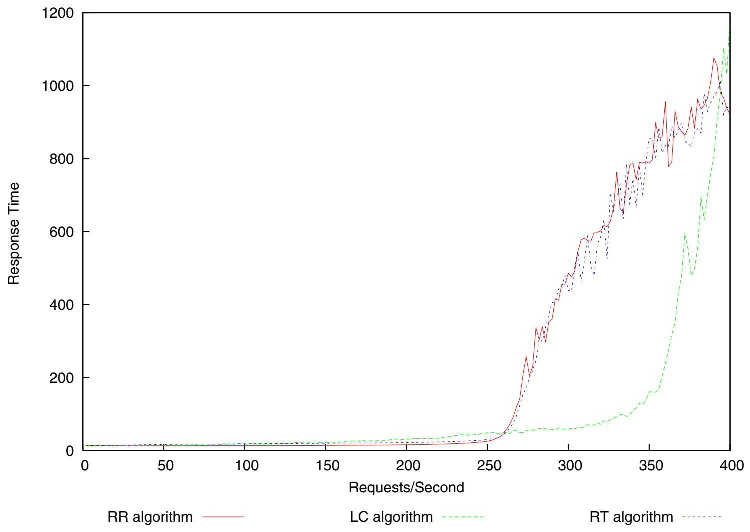


Fig. 7. Performance of the RT; LC and RR algorithms in an inhomogeneous environment. Surprisingly both RR and RT measurements behave similarly under load, while RR is best at low loads. This indicates that there is no advantage to using RT. LC holds out longest here before saturating as it is able to utilize the extra capacity in the faster servers.

having perfectly identical hardware is near zero given the rate at which technology advances. Equipment bought even a few weeks later could have dramatically different specifications.

To investigate the effects of inhomogeneities on the balancing, we manually adjusted the CPU speed on some of the blades to create a stepwise inhomogeneous environment with the following processor capacities: 1.4, 1.75, 2.1, 2.45, 2.8 GHz. The result is shown in figure 6 and 7.

Figure 6 shows results from using Least Connected (LC), Round Robin (RR), and Response Time (RT) algorithms to balance load between 5 servers. The algorithms perform approximately the same under low loads. The LC algorithm performs well marginally longer than the others. The RT algorithm perform almost as well, but succumbs to noise before earlier.

For the inhomogeneous case, the results are more surprising. Here the Round Robin and Response Time strategies (based on actual measurements of the system performance) perform equally well. Both succumb to noise at about the same point. Remarkably the Least Connections approach (which uses state information directly in the dispatcher) survives the longest. We suspect that this is because, once the variance in the computing nodes blows up due to thrashing, it dominates the measurements of response times and they become so unreliable that the information make no difference and the performance of RT. RT is therefore no better than RR (random chance) in practice. The dominance of uncertainty matches the conclusions made in [5]. The fact that Least Connections works longer is probably a result of the greater reliability of the state kept by the dispatcher combined with the fact that the faster machines finish their equal tasks more quickly, thus maintaining an equilibrium. In real traffic, job lengths will be variable and we suspect that LC will not perform significantly better than random chance.

9 Conclusions

There is surprisingly little literature on dispatcher load balancing, in spite of the importance of this technology for business. Ref. [10] performs similar studies to ours and arrives at somewhat different conclusions. In [10,11] they found the RR algorithm to be the worst under all conditions. Teo[10] also show results that indicate that RR, LC and RT converges at high loads. These results contradict our results and the ones found by Cardellini[12], as we found the RR algorithm to have best performance at low intensities, and that the LC algorithm could sustain higher traffic intensity than RR and RT. We suspect that the results are dependent on the nature of the test data downloaded. Our results have been easily controllable, and yet we see the effects of uncertainty start to dominate behaviour. In this regime, none of the expensive technology does much better than random chance.

In terms of scalability, we see that adding servers is not a smooth operation, in contrast with ref. [10]. Even at high traffic loads, modern servers have such power that they lead to very discontinuous changes in the performance. If one fixes the traffic level, the addition of a single server can have a large impact on the processing time, assuming that the trunk service is not a bottleneck. One can move from saturation to a comfort zone fairly easily. In this regard, there is no reason not to have extra capacity available in a server centre. Even a single extra server on standby can make a significant impact if one can predict the rise.

Our results indicate that there is little reason to use anything other than RR as a load sharing algorithm. This is a rather depressing conclusion. One would hope that monitoring and analysis would improve performance, but there is no strong evidence of this here.

What does the data centre engineer learn from this study? We believe that the key message of this work is that dispatcher capacity is the key in server load balancing. The sharing algorithm plays a lesser role than the ability of the dispatcher to cope with the connections. A simple Round Robin sharing works approximately as well the more sophisticated methods, but as the total load increases, the efficiency of the sharing appears to waver. We are not able to confirm this last conclusion in our experiment, but we think this point is worth further study. Once again, Service Level Objectives are far from predictable close to maximum capacity. We see here an operating level of 80% of maximum as being a reasonable over-capacity in production.

We are grateful to Jon Henrik Bjørnstad and Sven Ingebrigt Ulland for helpful discussion. This work is supported by the EC IST-EMANICS Network of Excellence. (#26854)

References

1. British Standards Institute. *BS15000 IT Service Management*, 2002.
2. J. Sauvé et al. Sla design from a business perspective. In *IFIP/IEEE 16th international workshop on distributed systems operations and management (DSOM), in LNCS 3775.*
3. W. Xu, X. Zhu, S. Singhal, and Z. Wang. Predictive control for dynamic resource allocation in enterprise data centers. In *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*, pages 115–126. IEEE Press, 2006.
4. X. Li, L. Sha, and X. Zhu. Adaptive control of multi-tiered web applications using queueing predictor. In *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*, pages 106–114. IEEE Press, 2006.
5. J.H. Bjørnstad and M. Burgess. On the reliability of service level estimators in the data centre. In *Proc. 17th IFIP/IEEE Distributed Systems: Operations and Management (DSOM 2006)*, volume submitted. Springer, 2006.
6. D.A. Menascé and V.A.F. Almeida. *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning.* Prenctice Hall, 2000.
7. M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems.* J. Wiley & Sons, Chichester, 2004.
8. G.B. Rodosek. Quality aspects in it service management. *IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, page 82, 2002.
9. A. Høyland and M. Rausand. *System Reliability Theory: Models and Statistical Methods.* J. Wiley & Sons, New York, 1994.
10. YM Teo and R Ayani. Comparison of load balancing strategies on cluster-based web servers. *Transactions of the Society for Modeling and Simulation*, 2001.
11. Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *SIGMETRICS '98/PERFORMANCE '98: Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 151–160, New York, NY, USA, 1998. ACM Press.
12. V Cardellini and M Colajanni. Dynamic load balancing on web-server systems. *Internet Computing IEEE*, 3(4):28–39, 1999.

Quantifying the Complexity of IT Service Management Processes

Yixin Diao and Alexander Keller

IBM T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598, USA
{diao, alexk}@us.ibm.com

Abstract. Enterprises and service providers are increasingly looking to process-based automation as a means of containing and even reducing the labor costs of systems management. However, it is often hard to quantify and predict the additional complexity introduced by IT service management processes before they actually have been deployed. Our approach consists in looking at this problem from a different, new perspective by regarding complexity as a surrogate for potential labor cost and human-error-induced problems: In order to effectively evaluate the benefits of IT service management processes – and to target the types of processes that contribute most to management complexity and cost – we need a set of metrics for quantifying the complexity and human cost of carrying out IT service management processes. This paper proposes such measures, and demonstrates how they can be applied to a typical service delivery process in order to assess its complexity hotspots as a basis for process re-engineering.

1 Introduction

The complexity of managing computing systems and information technology (IT) processes represents a major impediment to efficient, high-quality, error-free, and cost-effective service delivery, ranging from small-business servers to global-scale enterprise backbones. In order to accomplish these goals, enterprises and service providers turn increasingly to the IT Infrastructure Library (ITIL) [1]. ITIL comprises disciplines such as service management, support and delivery, and has established itself as the most widely used standardized process-based approach to IT service management. When implementing ITIL by means of IT management processes, however, one needs to be able to quantitatively measure the degree of IT management complexity exposed by particular processes, so that process designers and architects can discover complexity hotspots early in the design phase, and optimize the IT processes to reduce their complexity.

We regard complexity as a surrogate for potential labor cost and human-error-induced problems: IT systems and processes with a high degree of complexity demand humans and expertise to manage that complexity, increasing the total cost of ownership. Likewise, complexity increases the amount of time that must be spent interacting with a computing system or between administrators to perform the desired function, and therefore decreases efficiency and productivity.

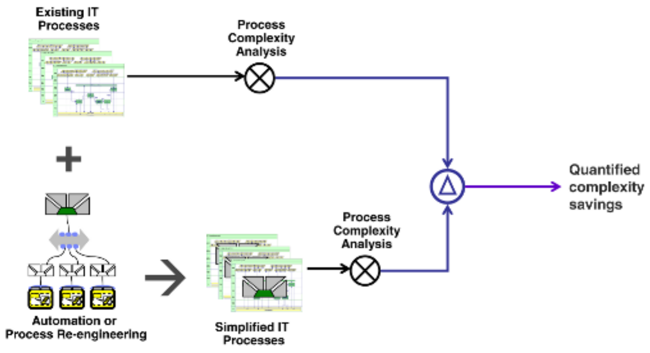

Existing IT Processes
Process Complexity Analysis
+
Quantified complexity savings
Process Complexity Analysis
Automation or Process Re-engineering
Simplified IT Processes

Fig. 1. Quantitative IT management process complexity evaluation

Furthermore, complexity results in human errors, as complexity challenges human reasoning and results in erroneous decisions even by skilled administrators. The goal is to reduce IT process complexity by designing, architecting, implementing, and assembling systems and processes with minimal complexity level.

There is little existing work in the area of process complexity analysis: Business process modeling tools typically include a process simulator, which allows the designer to run a set of simulated process executions (either through Monte Carlo or discrete event simulation) in order to assess the performance of the process, generate statistics about its execution, and pinpoint potential areas of improvement and optimization. However, many parameters need to be specified by the designer a priori: for example, for each task (or action), a duration is assigned during the modeling phase; for decisions, the designer needs to indicate up-front a percentage for each of the branches that indicates the probability that it will be taken (the sum of all branch probabilities equals 100%). None of the available business process model simulators focuses on process complexity as described in this paper. There is no overlap between existing work in the process simulation area and the work presented in this paper.

Related work in the system administration discipline has been carried out with a focus on establishing cost models, which take into account the impact of decisions. The most relevant work is [4], which generalizes an initial model for estimating the cost of downtime [7], based on the previously established System Administration Maturity Model [5]. Other related work can be found in information theory (e.g., Kolmogorov complexities [6]) or quality management in manufacturing (e.g., Six Sigma [9]).

Figure 1 illustrates our vision of quantitative IT management process complexity evaluation: Once a set of IT processes has been designed and architected, a process complexity analysis is carried out to pinpoint possible complexity hotspots and inefficiencies (depicted at the top of the Figure). In order to mitigate or even eliminate complexity hotspots, techniques like IT process re-engineering can be used to re-design the process(es); in addition, activities within a process that exhibit a very high degree of complexity are identified as candidates that should be further modeled and investigated. For example, they can be broken down to human comprehensible sub-tasks, which can be further supported by tooling or delegated to automation. In a third step, the process complexity analysis is applied to the simplified IT processes (depicted at the bottom of the figure). Finally, the results of the 'before' and 'after' analyses are evaluated side-by-side to obtain quantified complexity savings in order to measure the improvements.

The focus of this paper is the process complexity analysis because the process complexity model and its measures are the basis of both 'before' and 'after' analyses of the overall quantitative process complexity evaluation. The paper is organized as follows: Section 2 overviews our model for IT management processes. Section 3 details the complexity measures for quantifying processes and Section 4 describes the tooling we have implemented. In Section 5, we perform a process complexity analysis by applying our model. Our conclusions are contained in section 6.

2 IT Process Complexity Model

This section describes the proposed model of IT process complexity, which is used for computing the operational complexity of IT processes. Configuration management, change management, release management, and problem management are all examples of IT service management processes.

In [2], we have introduced a model for assessing configuration complexity. The intent of that model was to capture and quantify the complexity of a straight-line flow through a configuration procedure. Three different complexity measures have been identified, which are briefly summarized below in order to provide some background for the following discussion: *Execution Complexity* refers to the complexity involved in performing the configuration actions that make up the configuration procedure, typically characterized by the number of actions and the context switch distances between actions. *Parameter Complexity* is the complexity involved in providing configuration data to the computer system during a configuration procedure. *Memory Complexity* takes into account the number of parameters that must be remembered, the length of time they must be retained in memory, and how many intervening items were stored in memory between uses of a remembered parameter.

While the above three complexity measures are sufficient to capture the interactions of an administrator with a managed system at runtime, we need to extend this model to provide a quantitative assessment of IT management complexity that involves:

1. interaction between 2 or more roles in a process,
2. passing data in various formats (a.k.a., business items) between tasks, and
3. decision making among multiple roles.

Specifically, we model a process as a set of roles, each of which may participate in a set of tasks that either consume or produce business items. As depicted in Figure 2, IT management processes are modeled using the following three components: Roles, Tasks, and Business Items. This is consistent with

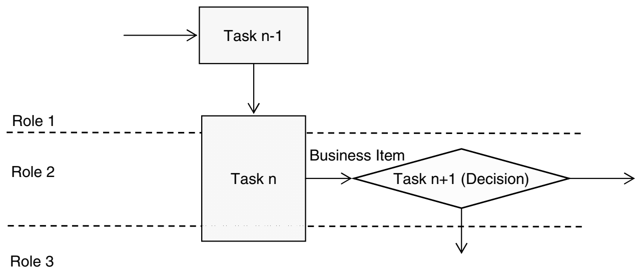


Fig. 2. Complexity model extensions for IT management processes

how common-off-the-shelf business process modeling tools, such as IBM WebSphere Business Modeler (WBM) or Tibco Business Studio structure business processes. Consequently, the information needed for complexity calculation can be extracted from typical process model data (see also section 4).

3 IT Process Complexity Measures

This section describes the per-task complexity metrics for each of the three aspects of the process complexity. Note that these metrics are designed to capture the first-order effects. However, our experience shows that even at this level they have demonstrated the effectiveness in identifying key automation opportunities and complexity bottlenecks, and in tracking process improvement.

3.1 Execution Complexity

Execution complexity covers the complexity involved in performing the tasks that make up the IT process. We use two metrics for execution complexity: base execution complexity and decision complexity.

Base Execution Complexity indicates complexity of the task according to its execution type. Values for this score are assigned according to a weighting scale of different task types. Each role involved in the task must be assigned an execution type chosen from below with corresponding values shown in square brackets next to the type name. The base execution complexity for that task is then the sum of values from all the roles. That is, for a task involving R roles ($r = 1, 2, \ldots, R$), its base execution complexity is computed as

$$E_{base} = \sum_{r=1}^{R} execType(r) \tag{1}$$

where the execution type $execType(r)$ is defined with regard to three types. *automatic* [0] - if the task is fully automated. *toolAssisted* [1] - if the task is manual, but tool-assisted. For example, manual triggering of a provisioning workflow or script involves providing workflow definitions. A service invocation task is also classified as toolAssisted, as it transfers the work to an external role. *manual* [2] - if the task is a fully manual procedure.

The above approach defines a normalized, unit-less score for base execution complexity. However, there are two possible extensions: (1) If data on average task times is available, they can be used to define the base execution complexity. (2) If task level procedure complexity analysis has been conducted, the base execution complexity can be defined with regard to procedure-wide execution, parameter, and memory complexity. Note that as a starting point we choose metric values in a linear scale (0, 1, 2) throughout the model; however, further studies need to be conducted to determine if a quadratic or exponential scale is more appropriate.

Decision Complexity quantifies additional execution complexity due to decision making. For non-decision making task, its value is zero. If a decision needs to be made, its complexity is based on the following four sub-metrics.

- $nBranches$: the number of branches in the decision. The intuition is that more choices result in higher decision complexity.
- $gFactor$: the degree of guidance. That is, how much information is provided to the user to make the correct choice. This is quantified with a three-level scale of increasing complexity: [0] for a specific (correct) recommendation of the decision branch to follow, [1] if general information is provided relating choices to goals so that a user could extract the correct decision with some processing of the information, and [2] if no information is provided to help the user select the correct choice.
- $cFactor$: the consequence of impact. That is, how significant is the impact if a wrong decision has been made. This is also quantified with a three-level scale of increasing complexity: [1] for negligible consequence, [2] for moderate consequence, and [3] for severe consequence.
- $vFactor$: the visibility of impact. That is, how much information is provided to the user to illustrate the consequences of their choice. This is also quantified with a three-level scale of increasing complexity: [1] for immediate consequence, [2] for short-term consequence, and [3] for long-term consequence in terms of end-state features.

With the above three sub-metrics, the decision complexity for that task is then multiplied by the number of roles (R) involved in that task.

$$E_{decision} = R \times (nBranches - 1) \times gFactor \times cFactor \times vFactor \qquad (2)$$

The above formula reflects the following intuition: the decision complexity is zero if the number of decision branches is one (no decision, straight flow); clear guidance reduces decision-making tasks to non-decision making tasks, even if it may involve multiple branches; immediate choice consequence may make the task tedious, if there are multiple branches and the guidance is unclear, but the level of complexity remains low since the decision uncertainty does not exist. Although the decision complexity formula may be expressed differently, we choose to use a multiplication format as we view the effects of multiple factors as being orthogonal to each other. For example, if the guidance is clear ($gFactor = 0$), the decision complexity is 0 no matter how many branches it may have. This is the case where the system administrator needs to classify problem tickets to many different categories such as 'file system full' and 'high application response time'.

3.2 Coordination Complexity

The per-task metrics for coordination complexity are computed based on the roles involved and whether or not business items are transferred.

Coordination Link Complexity is indicated by a unit-less value representing the complexity of coordinating between multiple roles. For each link between the task under consideration and other tasks carried out by different roles, score values are assigned according to a weighting scale of different coordination link complexities. The coordination link complexity for that task is then the sum of values from all the links ($l = 1, 2, \ldots, L$) multiplied by the number of roles (R) involved in that task.

$$C_{link} = R \times \sum_{l=1}^{L} linkType(l) \qquad (3)$$

where the link type $linkType(l)$ is defined as follows. *autoLink* [0] - if it is linking to an automated task. *controlLink* [1] - if it is a control flow link to a non-automated task without any business item being transferred. $dataTransferred$ [2] - if business items are transferred. *dataAdapted* [3] - if the transferred business items need to be adapted. For example, adaptation is needed if the consuming role is automated and the source data format is not machine-readable.

We also define **Shared Task Complexity** for tasks that involve multiple roles. For example, conducting a change review meeting requires the participation of multiple roles. The shared task complexity is computed from the assigned score below multiplied by the number of roles (R) involved in that task.

$$C_{task} = R \times taskType \times (meetingIndicator + 1) \qquad (4)$$

where *taskType* is defined as follows. *notShared* [0] - if it is not a shared task. *shared* [1] - if it is a shared task. *BIConsumed* [2] - if business items are consumed. The intuition is that coordination of data input requires extra cost. *BIProduced* [3] - if business items are produced. The intuition is that coordination of data output is more expensive because it requires agreement among multiple roles. The meeting indicator *meetingIndicator* takes a Boolean (0, 1) value: it takes a value of 0 if there is no meeting involved; otherwise, if a meeting is required, it takes a value of 1. Thus, having a meeting raises the shared task complexity by a factor of 2.

3.3 Business Item Complexity

The per-task business item (BI) complexity is computed based on the business items produced by the task under consideration.

Base BI Complexity is indicated by a unit-less value representing the complexity of involving business items. That is, for a task involving R roles and I business items either consumed or produced by that task, its base BI complexity is computed as

$$B_{base} = R \times I \tag{5}$$

BI source complexity is indicated by a unit-less value representing the complexity of supplying this field's value. Values for this score are assigned according to a weighting scale of different source complexities. Each field used in the business item must be assigned a source type chosen from one of the values given below. The field source complexity value for that field is then the value shown in square brackets below next to the type name. For each produced business item, a source complexity is assigned to each field based on the source that provides the field's data. Then, each source score is summed across the business items to produce the final per-task metric. That is, for a task involving R roles ($r = 1, 2, \ldots, R$), producing I_P business items ($i = 1, 2, \ldots, I_P$), and f fields ($f = 1, 2, \ldots, F_i$), its per-task business item complexity is computed as

$$B_{source} = R \times \sum_{i=1}^{I_P} \sum_{f=1}^{F_i} sourceScore(i, f) \tag{6}$$

where $sourceScore(i, f)$ is defined as follows: *internal* [0] - if the field value was produced from automation. $freeChoice$ [1] - if the field value can be chosen freely, e.g., a new password. *documentationDirect* [2] - if the field value was taken directly from the task documentation, an online source, or a process description manual (e.g., a Redbook), without extrapolation or adaptation. *documentationAdapted* [3] - if the field value was extrapolated from an example in the task documentation, an online source, or a process description manual. *bestPractice* [4] - if the field value would be obvious to a system operator versed in the administrative best practices for the application domain. *environmentFixed* [5] - if the field value is constrained by the environment to a specific value that is selected by the operator after further research. *environmentConstrained* [6] - if the parameter value is constrained by the environment to a limited set of possible choices.

These seven sources are ranked in order of increasing complexity burden. For example, a field sourced internally or pulled straight from the documentation does not require the system administrator to figure out its value. On the other hand, a parameter constrained by the environment, where that constraint is not obvious, imposes significant complexity since the system administrator needs to infer the possible legal value.

4 Process Complexity Model Tooling

We will now discuss the architecture, design and implementation of the tooling that we have developed in order to support the process complexity model and its measures that we have described in the previous sections. Our Integrated Complexity Analyzer is in charge of computing the complexity scores, according to the measures described in section 3.

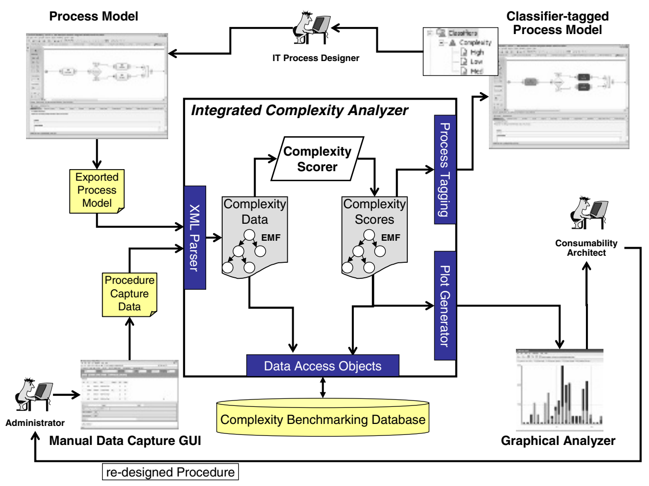


Fig. 3. Architecture of the Integrated Complexity Analyzer

4.1 Complexity Evaluation Scenarios

As depicted in Figure 3, there are two possible scenarios that need to be supported by the architecture of the Integrated Complexity Analyzer. The first scenario, depicted at the top of the figure, addresses complexity analysis at *process design time* and is the subject of this paper: An IT management process designer models a process by means of a common-off-the-shelf business process modeling tool, in our case IBM WebSphere Business Modeler version 6 [10]. In order to determine the complexity of the process and to pinpoint complexity hotspots, the designer exports the process to a file in the XML metadata interchange (XMI) format and executes our Integrated Complexity Analyzer. The latter automatically captures complexity data directly from the XMI file, computes the complexity scores and annotates the process model by tagging it with classifiers. The process designer can then re-import the process model and identifies, by means of color-coded activities that correspond to the different degrees of complexity, which activities and roles are considered complexity hotspots and should be simplified, if possible. Note that the redesign of the process model involves domain-specific knowledge and thus requires the involvement of the process designer. Once the model has been redesigned, the analysis is repeated until the complexity hotspots have been addressed.

The second scenario consists in having an administrator capture the configuration actions and parameters while executing a setup, change or configuration procedure for a product on a (distributed) system. This scenario, along with its measures, is the subject of [2]. It is depicted at the bottom of Figure 3. The major differences to the first scenario are as follows: First, the complexity analysis is performed at *runtime* and therefore reflects a straight-line flow through the procedure. In addition, all complexity data is gathered manually (by means of a web based manual data capture GUI) as no tooling is available that would log the actions automatically; furthermore, assigning complexity ratings to configuration parameters requires the involvement of an administrator. Finally, the consumers and producers of the complexity data are typically different people: while the administrator is needed to perform the configuration procedure and capture the procedure complexity data, the consumer of the complexity scores is typically a so-called consumability architect, whose role consists in diagnosing the configuration procedure for a specific product from a complexity perspective in order to re-design the procedure. Once the administrator has completed the procedure, the procedure capture data are input to the Integrated Complexity Analyzer, which calculates the complexity scores from the input data and displays the complexity scores on a per-activity basis by means of the graphical analyzer. This data can then be viewed and interpreted by the consumability architect of the product/procedure so that he can identify complexity hotspots and re-design the procedure, which is then again carried out by an administrator. The differences in the complexity scores for subsequent runs of the procedure reflect the quantitative improvements in terms of complexity.

4.2 Tooling Components

In addition to addressing the requirements for two fairly different usage scenarios, the architecture of the Integrated Complexity Analyzer needs to be adaptable to changes in the model and the summary scores. This is needed because the model is continuously being refined, based on the results we obtain by running a variety of scenarios. Traditionally, this is a major challenge, because the model and its measures are at the heart of the overall system, and any change to the model ripples through the data structures that are evaluated by the Complexity Scorer to compute the complexity scores. In order to mitigate the impact of changes, we have decided to isolate the representation of the data (both input and output data) as much as possible from the complexity scorer logic. A thorough modularization of the architecture and a combination of several technologies turned out to be particularly useful in order to accomplish this. We will describe each of the components of the Integrated Complexity Analyzer along with the technologies that have been used in their implementation.

We use an 'XML-centric' approach (as changes to the complexity model are introduced into the overall system by a change of the XML schema) and rely on the Eclipse Modeling Framework (EMF) [3] to automatically generate the Java objects that correspond to the elements in both XML schemas that represent the incoming complexity data and the complexity scores, respectively (depicted in the center of Figure 3). By doing so, every time a new element or attribute is added to an XML schema, we simply re-generate the EMF objects for the XML schema in which the change occurred. While a corresponding EMF object

– a Java class with accessor methods – representing the new XML element is generated (for which new code needs to be written in the complexity scorer), already existing EMF objects remain unchanged. An additional advantage of EMF is that we obtain the XML parser 'for free', as appropriate Java classes to serialize/deserialize XML files into/from EMF objects are automatically generated by EMF. Seamless transformation between XML, UML and relational database schemas based on Java is the core purpose of EMF.

The *Complexity Scorer* – the core component that summarizes and scores the complexities of both processes and procedures according to the measures described in the previous section – is implemented in Java and inputs the complexity data that is represented as an EMF model. The complexity scorer needs to distinguish between the two scenarios described above as their complexity data is fairly different. The XML schema for input data specifies a flag indicating whether the data in an input file refers to either procedure capture data or to an exported process model. The calculated complexity scores are represented as EMF objects, too.

The remaining three components of the Integrated Complexity Analyzer leverage various EMF extensions. The *Plot Generator* inputs an EMF model containing the complexity scores and transforms them into a graphical representation (typically a bar chart whose various display options can be selected by the user). Flexible support for graphic widgets is provided by the eclipse plugin for Scalable Vector Graphics (SVG) [11], which can be directly displayed in the Mozilla Firefox v1.5 web browser. The *Process Tagging* component works off the EMF model containing the complexity scores and serializes them by means of EMF into the XMI format so that the classifier-tagged process model is output in a format that can be directly consumed by the WebSphere Business Modeler tool. Finally, for each analysis, both the raw complexity data that is input into our system as well as the complexity scores computed by the Integrated Complexity Analyzer are persistently stored in a hybrid relational/XML database. We use an early adopter version of the IBM DB2 Universal Database Enterprise Server Version 9.1. The *Data Access Objects* implement persistent storage for EMF models in a relational database based on Service Data Objects (SDO) [8]. The advantage of storing structured EMF objects over storing a set of text documents merely as character large objects (CLOBS) in an RDBMS improves their retrieval significantly: as the data remains structured, one can easily issue queries of the type 'retrieve all the actions in a procedure/process whose memory complexity is greater than X' against the database.

5 Evaluation

In this section we consider a sample subprocess of ITIL Change Management: the process accepts a change request and updates the corresponding configuration items (CIs). A change request results in the creation of new CIs or the modification of existing CIs. As part of the process, it is necessary to extract information referring to the CI from the change request, authorize and validate the changes

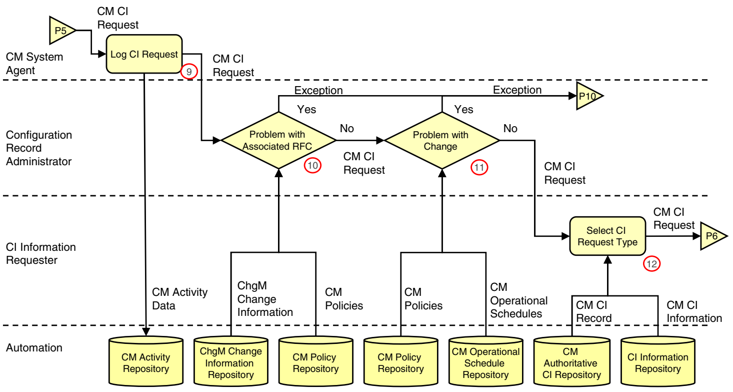


Fig. 4. Partial workflow for a change management subprocess

against policies, issue necessary queries to extract other needed CIs to accommodate the request, request modification of the CM Authoritative CI Repository design if needed (for new CI types), either create or update the appropriate CIs, and finally report the results of the CI changes in a history repository.

The sample subprocess consists of 27 tasks; a part of the flow is depicted in Figure 4, which includes the tasks numbered from 9 to 12 that are carried out by four different roles: Change Management (CM) System Agent, Configuration Record Administrator, Configuration Item (CI) Information Requester, and Automation (with a set of data repositories and services).

The per-task complexity is computed based on the complexity metrics introduced in Section 3. For example, task 11 is a decision point which checks if a requested change is allowed by the policies and execution schedule. It generates either an exception or proceeds to task 12 if the change can be carried out.

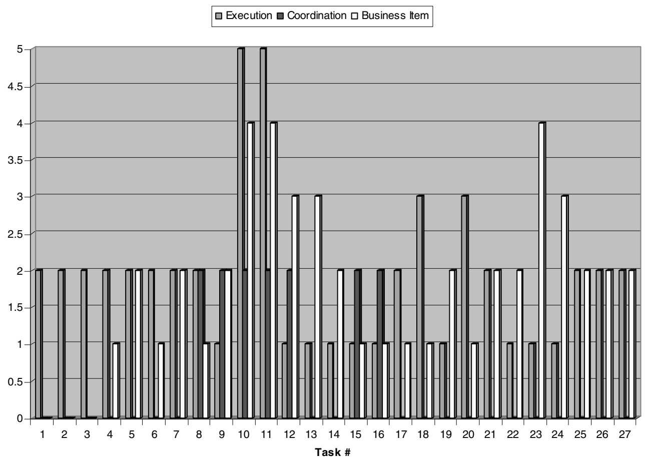


Fig. 5. Process complexity per-task view for a change management subprocess

Table 1. Process-wide complexity metrics

Complexity Measure	Metric	Value
Execution	Number of Tasks	27
	Number of Decision Points	11
Coordination	Number of Shared Tasks	0
	Number of Meetings	0
Business Item	Number of Business Items	10

Task 11 involves one role, four business items, two decision branches, and one coordination link with a different role (CI Information Requester).

Once the per-task metrics have been computed, they can be aggregated to produce process-wide views to identify the complexity bottlenecks within this process, or process-wide metrics to facilitate cross-process comparison. Per-task views are graphs showing all per-task metrics in bar charts. Figure 5 provides a per-task view for all 27 tasks. The x axis indicates the tasks, and the y axis indicates the metric values. All per-task metrics can be plotted separately, or aggregated for three high-level views of execution complexity, coordination complexity, and business item complexity. The per-task metrics can be analyzed to pinpoint complexity bottlenecks. For example, Figure 5 indicates that task 10 and 11 are relatively more complex in comparison to the other tasks in the process. On the other hand, this process does not have significant complexity spikes. The overall process complexity metrics are summarized in Table 1.

6 Conclusions and Outlook

In this paper we have proposed an approach to quantifying complexity of IT management processes. We model a process as a set of roles, each of which may participate in a set of tasks that either consume or produce business items. Moreover, process complexity is quantified from three dimensions: execution complexity with regard to the level of automation and decision making, coordination complexity with respect to the coordination links and the complexity of shared tasks, and business item (BI) complexity representing the source score for supplying values into the business item. Finally, we have described the design and implementation of the tooling we implemented, and have evaluated an IT management process to demonstrate the usage and applicability of the approach.

The benefits of our approach are as follows: First, the complexity analysis results guide the IT process reingeneering effort and identify activities that should be delegated to automation. Second, the IT process complexity model and its measures establish the basis for measuring improvements between two versions of an IT service management process. Third, the approach is applicable to different stages in the lifecycle of a process: while its core focus is on providing a quantitative framework for evaluating processes in the design stage, it can be applied to an already deployed process as well, in which case the measurements obtained during its execution apply to a specific flow through the process.

While these initial results are encouraging, there are several areas of further work: As an example, we are currently working on a mapping from the measures in this paper to higher-level measures such as success probability, configuration time, labor cost, and required skill level to complete configuration tasks. We are also exploring more service management process examples such as problem management and release management, and studying its applicability to business processes, e.g., Information and Communication Technology (ICT) business management processes for communication services, fulfillment, and billing.

Acknowledgments

The authors would like to express their gratitude to Aaron B. Brown for his work on an early version of the complexity model and to Christopher Ward for his assistance in obtaining a variety of IT Service Management Process Models. Both are employed by IBM. In addition, we are indebted to the reviewers for their helpful and constructive comments that helped us improve the quality of this paper.

References

1. IT Infrastructure Library. ITIL Service Support, version 2.3. Office of Government Commerce, June 2000.
2. A.B. Brown, A. Keller, and J.L. Hellerstein. A Model of Configuration Complexity and its Application to a Change Management System. In A. Clemm, O. Festor, and A. Pras, editors, *Proc. of the 9th IFIP/IEEE International Symposium on Integrated Management (IM 2005)*, pages 631–644, Nice, France, May 2005. IEEE.
3. F. Budinsky, E. Merck, and D. Steinberg. *Eclipse Modeling Framework*. Addison-Wesley, 2nd edition, 2006.
4. A.L. Couch, N. Wu, and H. Susanto. Toward a Cost Model for System Administration. In D.N. Blank-Edelman, editor, *Proc. 19th Large Installation System Administration Conference (LISA '05)*, pages 125–141, San Diego, CA, USA, December 2005. USENIX.
5. C. Kubicki. The System Administration Maturity Model – SAMM. In *Proc. 7th Large Installation System Administration Conference (LISA '93)*, pages 213–225, Monterey, CA, USA, November 1993. USENIX.
6. Ming Li. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 1997.
7. D. Patterson. A Simple Way to Estimate the Cost of Downtime. In A.L. Couch, editor, *Proc. 16th Large Installation System Administration Conference (LISA '05)*, pages 185–188, Philadelphia, PA, USA, November 2002. USENIX.
8. Service Data Objects. `http://www.eclipse.org/emf/sdo.php`.
9. Geoff Tennant. *Six Sigma: SPC and TQM in Manufacturing and Services*. Gower Publishing, Ltd., 2001.
10. IBM WebSphere Business Modeler. `http://www-306.ibm.com/software/integration/wbimodeler`.
11. World Wide Web Consortium, W3C Recommendation. *Scalable Vector Graphics (SVG) 1.1 Specification*, January 2003. `http://www.w3.org/TR/SVG/`.

Ontology-Based Knowledge Representation for Self-governing Systems

Elyes Lehtihet[1,3], John Strassner[2], Nazim Agoulmine[3], and Mícheál Ó Foghlú[1]

[1] Telecommunications Software & Systems Group - Waterford Institute of Technology
Waterford, Ireland
{elehtihet, mofoghlu}@tssg.org

[2] Autonomic Research Lab - Motorola Labs,
Schaumburg, IL 60010, USA
John.Strassner@motorola.com

[3] Networks and Multimedia Systems Group - University of Evry-Val d'Essonne
Evry Courcouronnes, France
Nazim.Agoulmine@iup.univ-evry.fr

Abstract. Self-governing systems need a reliable set of semantics and a formal theoretic model in order to facilitate automated reasoning. We present an ontology-based knowledge representation that will use data from information models while preserving the semantics and the taxonomy of existing systems. This will facilitate the decomposition and validation of high level goals by autonomous, self-governing components. Our solution reuses principles and standards from the Semantic Web and the OMG to precisely describe the managed entities and the shared objectives that these entities are trying to achieve by autonomously correlating their behavior. We describe how we created UML2, MOF, OCL and QVT ontologies, and we give a case study using the NGOSS Shared Information and Data model. We also set the requirements for integrating existing information models and domain ontologies into a unique knowledge base.

1 Introduction

The representation of knowledge at the autonomic manager level is a critical issue for designing and deploying self-governing systems. We see an autonomic manager as a set of software agents that use an expressive and dynamically updateable knowledge base to represent the relationships between managed entities, the system configuration, the objectives of the administrators (policies) and the explicit semantics of the goals of the system. For autonomic networks, this flexible knowledge base enables the system to dynamically adjust to the changing demands of its users, as well as changing environmental conditions.

Self-governing systems can be described as very reactive systems, with a precise modeling structure, data description and behavior. The basis for structuring is hierarchical decomposition and type hierarchies. Data description is based on data types of values and objects. The basis of behavior description is extended finite state machines communicating by messages [1].

A knowledge base requires an expressive, unique representation. Arguably, this is best realized using a knowledge representation language with well defined semantics and a practicable inference algorithm for reasoning on a system specification. A system specification, in a broad sense, is the specification of both the behavior and a set of general parameters of the system [1].

The integration of the structural and the dynamic aspects of a self-governing system is a difficult task that is even harder to achieve because of: 1) the complexity of describing and refining a high level goal into lower level objectives that can be enforced by autonomous system entities, and 2) the lack of interoperability between existing modeling standards. Interoperability is a major issue for system integrators and tools vendor; several initiatives are trying to solve the problem by unifying knowledge representation without loosing the associated semantics of the data. Until this is accomplished, it will be impractical to effectively share models among heterogeneous (competing) modeling tools.

In this paper, we present an approach for unifying the representation of the information needed by an autonomic manager, and preserving the inherent semantics of a domain — this is a crucial step toward automating the reasoning process and integrating heterogeneous knowledge bases.

1.1 Using Software Engineering Principles to Specify Self-governing Systems

An autonomic manager is a software agent, which must be developed by applying technologies and best practices from computer science, applications domains and other fields. Due to the complexity of describing a high level goal and its refinement into low level objectives, we argue that an agile (adaptive) development framework, as opposite to a plan-driven (predictive) framework, is the most appropriate methodology for designing, deploying and testing self-governing systems. The Agile methodology is quite simple to understand: *Instead of creating extensive models before writing source code, you start by developing and testing agile models with a concise and a precise kernel.*

Another approach for defining IT system specification is the OMG Model Driven Architecture (MDA). The aim of MDA is not to replace the existing system with something completely different; the objective is to make that system more efficient by incrementally automating the parts of it that can easily be automated, so it will accelerate the integration of new technologies and automate their integration into the existing solution.

Many would argue that principles of Agile Modeling and OMG Model Driven Architecture are disjointed (not compatible), but we see the two methodologies as complementary: First, to accelerate the (agile) development, the behavior of a system must be derived from the specification. It is only this way that we will ensure that the business goals (desired behavior) meet the implementation (effective behavior). Second, as software architectures grow in size and complexity, the need to subsequently incorporate software models into the system development automation stream grows even faster.

In the white paper comparing the OMG-MDA and the TMForum Next Generation Operation Support System (NGOSS) [2], the authors argue that: *"While following an approach similar to MDA, NGOSS has chosen to focus on building a framework for identifying and specifying well-defined business and system views on a modeled OSS/BSS solution"*. NGOSS principles and standards will be discussed more in detail in section 4 and 5. Note that our work extends the work of NGOSS.

The OMG Unified Modeling Language (UML) is the software industry's dominant modeling language. The current standard is UML 2.0., a major rewrite on the previous 1.5 version. This 1.5 version, used by the NGOSS Shared Information and Data model (SID), will continue to be the official current version until all four components of UML2 (Infrastructure, Superstructure, Diagram Interchange and Object Constraint Language) are completed and ratified.

In section 8.2 of the OMG Ontology Definition Metamodel (ODM) specification [3], it is mentioned that: *"The lack of reliable set semantics and model theory for UML prevents the use of automated reasoners on UML models. Such a capability is important to applying Model Driven Architecture to systems integration ... UML lacks a formal model theoretic semantics, OCL also has neither a formal model theory nor a formal proof theory, and thus cannot be used for automated reasoning (today)"*.

The same arguments can be used against the OMG Meta Object Facility (MOF), and since every modeling language used in MDA "must be" described in terms of the MOF language, the OMG Vision is missing a formal, explicit specification of concepts and relationships for its modeling languages (UML2, MOF, OCL and QVT). Thus, any information model that "only" relies on UML would not fulfill the requirements of a decidable knowledge base and will prevent the use of automated reasoning.

Reasoning on a precise and computer-processable semantic is the ultimate objective of the Semantic Web vision. The essential principles, standards and technologies are discussed in the following section.

1.2 Semantic Web Technologies

While trying to ensure the long term growth of the web, the World Wide Web Consortium (W3C) issued the Web Ontology Language (OWL) — a markup language for publishing and sharing data using ontologies on the Internet. Ontologies are agreements about shared conceptualization [4], and hence a basis for information exchange by putting documents with machine-readable meaning (semantics) on the Web. OWL permits varying degrees of reasoning depending on the expressivity of the Description Logic subset that is used. Description Logics, sometimes called terminological systems or concept languages, are a family of knowledge representation languages which can be used to represent the terminological knowledge of an application domain in a structured and formally well-understood way [5].

OWL$\mathcal{DL}$ is based on the description logic category known as $\mathcal{SHOIN}(\mathcal{D})$. Its subset OWL Lite is based on the less expressive logic category $\mathcal{SHIF}(\mathcal{D})$. OWL

Full is the most expressive level but, because of that, *can* lead to infinite loops — not recommended for automated reasoning. The Semantic Web Rule Language (SWRL) extends the set of OWL axioms in order to include conditional rules (Horn clauses).

As noted by Ushold and Menzel in [4], the strength of the W3C standard for representing ontologies, in addition to its soundness and unique implementation, is the ability to express logical equivalence and other relationships between concepts, properties and individuals in different ontologies. One main weakness is the lack of support for procedural functions (e.g. arithmetic, string manipulation/comparison) that are, in our opinion too, essential for mapping between real-world ontologies.

The following section will give a summary of existing approaches for representing knowledge for Autonomic systems. In section 3, we describe the methodology that we have used to create a precise modeling language for information models. Then, we will demonstrate how our solution can be applied to the TMF NGOSS set of principles. Finally, we will discuss the results of our experiments and describe the future orientation of our research.

2 Related Works

As stated in [6], Information models alone are not enough to capture the semantic and behavior of managed network entities. The ideal solution is to use an ontology language to precisely formalize a domain problem. In this section, we will give a short overview on existing approaches and their use of information modeling.

2.1 Autonomic Network Management

In 2004, IBM issued a toolkit that includes components, tools and scenarios for designing and deploying Self-managing systems. In [7], the importance of ontologies in the design and the implementation of autonomic computing systems is described: *without an explicit meaning, the resolution of a problem is not possible.* As shown in Figure 1, IBM uses the DMTF Common Information Model (CIM) as a reference model to infer properties about distributed systems. The system behavior is expressed using the Simplified Policy Language (SPL). Common Base Event (CBE) is the IBM standard for exchanging messages between autonomic management engines (implementation of the autonomic-manager). The CBE model provides a basis for sounder problem determination and is a cornerstone of automatic computing system management. This profusion of formats creates a semantic gap between the specification of the system and its behavior; since they are not expressed in the same language and do not share the same semantic, how can they be integrated in a unique knowledge base? Furthermore, as noted in [6], the DMTF, IETF and ITU do not produce UML compliant models, and thus cannot reuse off-the-shelf UML tools to represent their concepts.

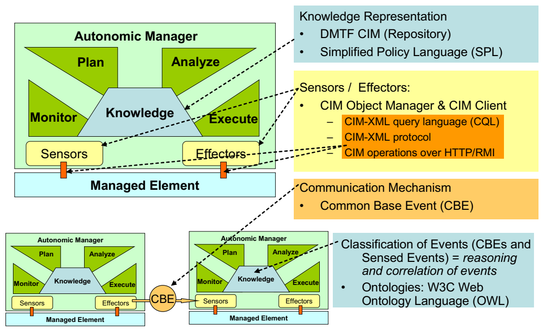

Autonomic Manager
Plan
Analyze
Monitor
Knowledge
Execute
Sensors
Effectors
Managed Element
Knowledge Representation
• DMTF CIM (Repository)
• Simplified Policy Language (SPL)
Sensors / Effectors:
• CIM Object Manager & CIM Client
– CIM-XML query language (CQL)
– CIM-XML protocol
– CIM operations over HTTP/RMI
Communication Mechanism
• Common Base Event (CBE)
CBE
Classification of Events (CBEs and Sensed Events) = reasoning and correlation of events
• Ontologies: W3C Web Ontology Language (OWL)

Fig. 1. IBM Autonomic Computing Vision

As shown in Figure 2, the advantage of using UML notations is to capture the specification of the system and its behavior. We believe that the implementation of Self-governing systems will rely on a specification of Executable Models [8] — with a formal model theoretic and a precise semantic, necessary to implement a reliable model compiler (Reasoner). This model compiler will allow automated reasoning and help in discovering hidden inconsistencies in the system specification.

Guerrero et al., in [9], proposed the utilization of OWL+SWRL for the definition of the management behavior. SWRL would replace the conventional policy language while the CIM-to-OWL mapping will enable system properties to be inferred. However, we think that this approach has two important limitations:

1. The fundamental difference between Object Oriented (OO) models and Ontologies is the representation of semantics: Ontologies use a declarative approach (Constraints, Axioms and Rules), but OO models only represent imperative semantics (operations). Therefore, it is not possible to map OO models into OWL without loosing semantics (e.g., for the CIM, as used in the IBM toolkit, semantics attached to the CIM Methods and Qualifiers are lost).
2. The DMTF uses their own proprietary Managed Object Format (DMTF-MOF), which is not compatible with the OMG-MOF. Hence, CIM Models will *not* produce valid UML models and, as discussed in section 1.1, it will not be possible to reuse any software development methodology (e.g., Agile or MDA) for the design, deployment and testing of a self-governing system.

In [10], López de Vergara proposed to refine and extend the CIM Metaschema by using the Object Constraint Language (OCL); however, this necessary formalization was not incorporated into the DMTF specification. Recently, the DMTF realized the advantage of aligning its model with the OMG standards. This key work will enable the use of off-the-shelf UML tools for CIM development. However, the first draft will not be available before the 3^{rd} quarter of 2006 [11]. Until then, we will not consider any CIM-based approach as a possible solution for representing knowledge for self-governing systems.

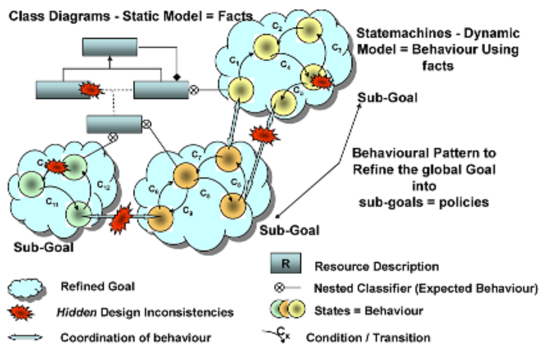

Class Diagrams - Static Model = Facts
Statemachines - Dynamic Model = Behaviour Using facts
Sub-Goal
Behavioural Pattern to Refine the global Goal into sub-goals = policies
Sub-Goal
Sub-Goal
Refined Goal
Hidden Design Inconsistencies
Coordination of behaviour
Resource Description
Nested Classifier (Expected Behaviour)
States = Behaviour
Condition / Transition

Fig. 2. Systems specification using UML

2.2 Ontologies, Information Models and Constraint/Rule Languages

Ontologies and Information models (Object Oriented Languages) have very similar approaches for the declaration of static structures, namely classes (concepts), class hierarchies (using inheritance), attributes, relationships, and instances [12]. However, an ontology only describes concepts and their inter-relationships; it does not provide support for behavioral features (e.g., operations, parameters, and state machines). Therefore, trying to represent a UML-based information model in OWL will lead to major inconsistencies and loss of valuable semantics.

As previously noted, SWRL extends OWL with Horn-like Clauses. SWRL provides formal semantics and thus allows computation, unlike the OMG-OCL which does not provide a formal proof theory (see section 1.1). We are working on a mechanism to map existing OCL Expressions into SWRL axioms, but this is beyond the scope of this paper and hence, will be developed in future work.

An advantage of using an ontology, in addition to the computation guarantee, is the uniqueness of the representation, since the implementation of the W3C specification guarantees interoperability between different ontology tools and repositories. On the other hand, for all OMG specifications, the informal definition of the concrete syntax is not given in the semantics document, but in the notation guide. There is no mapping between the concrete syntax and the abstract syntax. This lead to an implementation problem: there is no way to check that the output of a tool conforms to the language specification (UML, MOF, OCL or QVT). This is because the OMG only produces the specifications — documents that precisely describe what something should do, and how it should act. The implementations of the specifications (UML Modeling tools, Transformation Engine, Model Compiler/Checker) are not, and will never be, produced by the OMG [13]. Thus, the lack of constraints in the specification of the language (metamodel) and the heterogeneous implementations create informal (not precise) models.

We investigated the work of Cranefield [14] and Knublauch [12], who specified a mapping from UML models to RDF and OWL, respectively. However, these approaches are not appropriate for capturing the semantics attached to behavioral diagrams, as well as some semantic aspects of the class diagrams.

3 Ontology Based Metamodel for UML2, MOF, OCL and QVT

Our approach is to represent the UML, MOF, OCL and QVT *metamodels* into OWL models. Therefore, any UML *model* can be checked against the precise specification of its metamodel and thus, all the concepts from our system model (structure and behavior) will be instantiated in a unique format that will surely comply to the OMG specification previously captured by an ontology.

In this way, we have built a knowledge base using an expressive and unique language (OWL) with well defined semantics (Description Logic) and practicable inference algorithms ($\mathcal{DL}$ Reasoners) for reasoning on a system specification (UML structural and behavioral models).

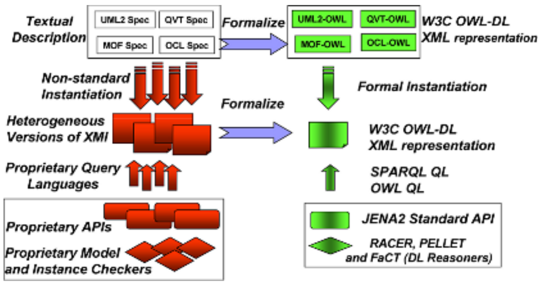

Textual Description
UML2 Spec
QVT Spec
MOF Spec
OCL Spec
Formalize
UML2-OWL
QVT-OWL
MOF-OWL
OCL-OWL
W3C OWL-DL XML representation
Non-standard Instantiation
Formal Instantiation
Formalize
Heterogeneous Versions of XMI
W3C OWL-DL XML representation
Proprietary Query Languages
SPARQL QL
OWL QL
Proprietary APIs
Proprietary Model and Instance Checkers
JENA2 Standard API
RACER, PELLET and FaCT (DL Reasoners)

Fig. 3. Ontology-based knowledge specification of reactive systems

3.1 Transformation Principles

We have implemented transformations from the main OMG specifications (MOF, UML2, OCL and QVT) to W3C-OWL. UML2, MOF1.4 and MOF-QVT are available in Rational Rose format on the OMG website. UML and MOF reuse the InfrastructureLibrary; MOF and OCL are decomposed into two main packages: Essential and Complete, while QVT extends EssentialMOF and EssentialOCL. However, we were not able to find any reference to CompleteOCL. As a result, every OMG sublanguage is a distinct package that has dependencies (import/merge) with other packages.

In Figure 4, we show an example of dependencies between UML top level packages and the UML subset that the SID Business View employs. In this paper, we outlined the transformation rules and detail the implemented parser for the XMI output of the Rational Rose Unisys Add-In. Every package was transformed into a separate OWL ontology (file).

For UML2, we generated 82 ontologies that import each others depending on their dependencies (import/merge) and the inter-references between elements in

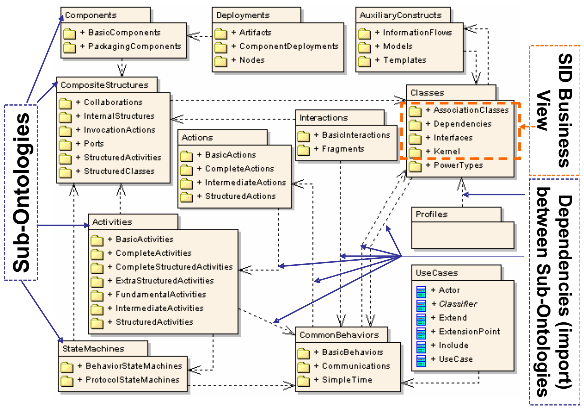


Fig. 4. UML2 top level packages and the SID Business View Subset

separate packages. Every class, owned by a package, has a unique name and thus map to an `owl:Class`. For the range of the attributes (String, Boolean, Integer and UnlimitedNatural), we did not want to use the predefined xsd datatypes (used by Protégé), the reason is that they are not supported by the current version of reasoners — this limitation should be addressed in the next version of OWL. Therefore, all the data types were represented as an `owl:Class`; the attributes were all mapped to `owl:ObjectProperty` with the following pattern : `<ClassName>.<AttributeName>`. The same mechanism was used for the AssociationEnds, where the role name of every *navigable* association end was mapped to an object property of the source class. We treated the cardinalities of the object properties, respectively `lower` and `upper`, as follow:

- [1..1] ⇒ `FunctionalProperty`;
- [0..n] ⇒ Default Cardinality;
- [x..n] ⇒ `minCardinality(x)`;
- [0..x] ⇒ `maxCardinality(x)`;
- [x..x] where x≠1 ⇒ `cardinality(x)`;
- [x..y] where x>0 ∧ x≠y ∧ y≠n ⇒ `minCardinality(x)` ∧ `maxCardinality(y)`.

The Enumeration Classes (AggregationKind, VisibilityKind, etc.) were mapped to an `owl:Class`. Their possible values, previously represented as attributes, were mapped to OWL instances of the owning class and explicitly made disjoint. Example: UML VisibilityKind enumerated attributes {package protected private public} were made disjoint by applying an `owl:allDifferents` construct.

We created our specific URI to identify the ontologies, which is also their online repository. The results of the transformations can be found at : http://www.tssg.org/public/ontologies/*org*/*spec*/*year*/*PackageName*.owl; where:

- UML2 : BaseUri + omg/uml/2004/UML2-Super-MDL-041007.owl
- QVT : BaseUri + omg/qvt/2005/QVT.owl
- MOF : BaseUri + omg/mof/2004/MOF.owl

The ontologies (which import their dependent ontologies), can be loaded in Protégé. When performing the Classification Hierarchy, the Reasoners will remove the unnecessary superclasses. For example, *Transition* and *RedefinableElement* are subclasses of *NamedElement*; and *Transition* is also a subclass of *RedefinableElement*. Therefore, the inheritance between *Transition* and *NamedElement* is superficial and thus can be removed without altering the consistency of the model — This should be considered as an optimization of the language.

3.2 Limitation of the Transformation

As noted in section 2.2, ontologies do not support behavioral features; thus the operations (and parameters) present in the UML2 metamodel were mapped into annotation properties (`rdfs:Comment`) of the owning class. UML2 encloses 107 private operations. All the operations specify an OCL expression as Text — not XML constructs. The automated mapping of OCL to SWRL is not supported by our tool as yet, a solution to this will be proposed in our future work.

OWL does not apply the Unique Name Assumption (UNA) by default: every concept is not, by default, necessarily distinct from the others. In our case, this implies that all the constructs of the OMG metamodel are not necessarily distinct. This in turn implies that an `Association` is not by default different from a `Class`. To precisely describe the metamodel, we implemented an automatic generation of disjointment between classes in the same package; however, we had to abandon this solution because it created too many inconsistencies related to multiple inheritances. Example: an `AssociationClass` is a `Class` and an `Association` at the same time, so if a `Class` is semantically disjoint from an `Association`, it implies that an `AssociationClass` is disjoint with itself. However, this can be solved in the instantiation of the model by explicitly creating an `owl:allDifferents` construct between instances or specifying an `owl:disjointWith` between subclasses.

There is another issue with the cardinality restrictions. Because ontologies use the Open World Assumption (OWA) for reasoning, it means that what is given to the reasoner is not necessarily complete. The reasoner will only generate an error when there are more than the allowed `owl:maxCardinality` (or `owl:cardinality`) instances associated with an object property. However, if there are fewer instances than the `owl:minCardinality` restriction then the reasoner will assume that it could be defined elsewhere and therefore infer that the ontology is consistent. The solution is to implement a pre-compiler to check the cardinalities and corresponding instances before using the reasoner to check to overall consistency of the model.

Another limitation concerns Package naming. There are only two cases where the name of the Package is duplicated in the specification. An algorithm can look for such conflicts and change the name of the package (sub-ontology); we used the convention `<owningNamespace>.<PackageName>`. Example: `InfrastructureLibrary.`*Profiles* and `UML.`*Profiles*.

4 Mapping the TMF NGOSS SID to OWL

Once the UML2 ontology was defined, we could check the consistency of any UML2 model (structure and behavior) against the precise specification of the language. In this section we will describe how we applied a mapping from the SID (UML Model) to an Ontology, without loosing the semantics of the modeling language (behavioral features).

As noted in section 1.1, the SID uses UML version 1.5. After a review of the UML specifications, we noticed that the main differences between the two versions concern the behavioral aspect of the language. For the object structure, with the exception of the NestedClassifier pattern that is not used in the actual version of the SID Business View, there is no difference between UML 1.5 and 2.0. Therefore, the UML subset used by the SID is fully compatible with the UML2 ontology since it does not use any behavioral diagram. The transformation rules are described in the following paragraphs.

The SID is the *"lingua franca"* for all TMF work. It defines a common set of concepts, in the form of an object-oriented information model, that all other TMF programs can use. The mapping from the SID to OWL, having a UML2 ontology, is quite straightforward. Every language construct used by the model: *Class*, *Association*, *Property*, *Operation*, *Attribute*, *Stereotype*, *DataType*, *AssociationClass*, *Dependency*, etc. has a direct equivalence in the ontology model. The interrelation between entities, *ownedAttributes*, *ownedOperations*, *ownedParameters*, *stereotypes*, *datatypes*, etc. is inherently present and constrained (cardinalities, domain and range) in the imported UML2 sub-ontologies: *Kernel*, *AssociationClasses*, *Dependencies* and *PrimitivesType*.

There are two different ways of representing a UML model: by instantiating the element of the model from their specification in the ontology (i.e., represent the model as a set of individuals), or by subclassing every element of the model and expressively constraining the values of its object properties (`owl:allValuesFrom`). The choice will not make any semantic difference for the reasoner. However, we preferred the second approach, as it offers a better visualization with the Protégé editor.

If the approach for representing the model is to subclass and constrain the corresponding language constructs, then all the `owl:Class` elements in a sub-ontology must be disjointed. But, if the representation mechanism is to create an instance of the model, then all the individuals must be distinguished by adding an `owl:allDifferents` axiom.

The mapping principles are also quite simple; every SID package will be represented in a separate file and imports the required UML2 sub-ontologies, in addition to the other SID packages referenced by its elements (datatype, constrainedElement, etc.). Every sub-ontology extend the imported *Kernel:Package* to create an `owl:Class` with the name of the package as an ID. The same mechanism is applied to every language construct. The algorithm for naming the sub-ontologies is similar to the one described in section 3.1.

One minor limitation is the representation of the *AssociationEnd* and *Constraint*. We implied that every element in the model must be properly named, because it is more readable than the *xmi:id* that distinctively identify the elements. However, it is not the case with all the SID elements, so we had to manually update the model.

5 Discussion and Future Work

Our approach can be used for any UML2-based model, and the UML2 ontology could be refined/extended to capture additional artifacts not present in the actual specification. Two important aspects of the NGOSS Methdology that can benefit from our solution are the NGOSS Contracts and the NGOSS Metamodel specifications.

NGOSS Contracts extend the *Design by Contract* paradigm to ensure that all information that is exchanged between components is done so in a consistent way. All NGOSS Contracts have a view-specific portion (NGOSS contains views that are used to represent the needs of different constituencies, such as business analysts vs. programmers and architects). The view specific model part contains various types of models (UML and others) tailored to support the specific view of the contract, i.e. Business, System, Implementation and Deployment views. All these views need to be integrated in order to provide a coherent mapping between NGOSS views: reasoning on Contracts models. Therefore, our solution could be fully reused to represent the model part of the Contracts expressed at different level; the Reasoning mechanism would allow their automatic validation.

The NGOSS Metamodel (TMF053D) extends the UML metamodel in order to introduce specific concepts, building blocks and artifacts that are required to represent telecom needs. Thus, TMF053D is a necessary reference document to support the creation of NGOSS-based models of software system solutions. These models will capture specific aspect of the telecom world and will need to be stored in a unique and precise knowledge base. Now, for the same reason described in section 2.2, a solution based on the extension of the UML2 Ontology with the NGOSS artifacts will ensure the different NGOSS stakeholders that the semantic of their systems will be fully captured in the model and they will have a unique, open repository for all their views of the system based on the "homogeneously" implemented W3C OWL specification.

6 Conclusion

Autonomic systems require knowledge from different sources to be represented in a common way. While conflicting attribute and datatype definitions present problems, semantic dissonance is a far more difficult problem to solve — one that requires an extensible, common representation of knowledge that does not lose its associated semantics. This paper has introduced an ontology-based knowledge representation to solve this problem. We have used the algorithms described in this paper to construct an OWL representation of the TMF NGOSS SID, which we are using in other autonomic computing works. The OWL mapping provides a *machine-readable* representation of the SID managed entities and concepts

— this enables semantics to be properly captured and associated with model elements.

Future work will include mapping OCL into SWRL axioms and building a reasoner that can use the OWL SID mapping (which includes NGOSS Contracts) to build a reliable reasoner for contract-based interactions and workflows. These two work items will then be used to continue our research in semantics for autonomic computing.

Acknowledgement

This research has been funded by SFI *"Autonomic Management of Communications Networks and Services"* PI Cluster Award: 04/IN3/I404C.

References

1. International Telecommunication Union (ITU-T): Specification and Description Language (SDL), Recommendation Z.100, August 2002.
2. Strassner J., Fleck J., Huang J., Faurer C., Richardson T.: TMF White Paper on NGOSS and MDA, TMForum, April 2004.
3. Object Management Group (OMG): Ontology Definition Metamodel, Fourth Revised Submission, November 2005.
4. Ushold M., Menzel C.: Achieving Semantic Interoperability & Integration Using RDF and OWL, W3C Draft, January 2006.
5. F. Baader, D. Calvanese, D. L. McGuiness, D. Nardi, P. F. Patel-Schneider: The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press, Cambridge, UK, 2003
6. Strassner J., Agoulmine N., Lehtihet E.: FOCALE A Novel Autonomic Networking Architecture, in Latin American Autonomic Computing Symposium (LAACS), July 18-19, 2006, Campo Grande, MS, Brazil.
7. Stojanovic L., Schneider J., Maedche A., Libischer S., Studer R., Lumpp Th., Abecker A., Breiter G., Dinger J.: The role of ontologies in autonomic computing, Published in IBM Systems Journal, Volume 43, Issue 3, 2004.
8. Mellor S. J., Balcer M. J.: Executable UML: A Foundation for Model Driven Architecture, Addison-Wesley Longman Publishing Co., Inc., 2002.
9. Guerrero A., Villagrá V. A, López de Vergara J. E, Berrocal J.: Ontology-Based Integration of Management Behaviour and Information Definitions Using SWRL and OWL. DSOM 2005, October 24-26, 2005, Barcelona, Spain: pp 12–23.
10. López de Vergara J. E, Villagrá V. A, Berrocal J.: On the formalization of the Common Information Model metaschema. DSOM 2005, October 24-26, 2005, Barcelona, Spain, pp 24-26.
11. DMTF Newsletter : can be found at `http://www.dmtf.org/newsroom/newsletter/2006/05/page4`, May 2006.
12. Knublauch H. : An Agile Development Methodology for Knowledge-Based Systems Including a Java Framework for Knowledge Modeling and Appropriate Tool Support, Dissertationsschrift (PhD thesis), University of Ulm (2002)
13. Object Management Group - Specifications and Process, can be found at `http://www.omg.org/gettingstarted/specsandprods.htm#SpecProd`, May 2006.
14. Cranefield S. : Networked Knowledge Representation and Exchange using UML and RDF, Journal of Digital Information, Volume 1 Issue 8 Article No. 44, 2001.

An Ontology-Based Approach to the Description and Execution of Composite Network Management Processes for Network Monitoring

José María Fuentes, Jorge E. López de Vergara, and Pablo Castells

Departamento de Ingeniería Informática, Escuela Politécnica Superior,
Universidad Autónoma de Madrid, Campus de Cantoblanco, 28049 Madrid, Spain
{Chema.Fuentes, Jorge.Lopez_Vergara, Pablo.Castells}@uam.es

Abstract. Web service technology has been proposed to implement management interfaces of managed resources. These web services can usually be combined to perform composite processes. These composite processes can be defined with service ontologies such as OWL-S, which allows their formal description. However, other technologies, including the Web Services Business Process Execution Language (WSBPEL), provide more mature execution engines. This paper presents an approach to define and execute composite network management processes with existing technology. For this, a use case is developed in which a set of web service interfaces are defined for a network probe, and a composite process is specified using OWL-S to monitor the network load. Then, this specification is later translated to WSBPEL and interpreted by a real execution engine.

Keywords: OWL-S, WSBPEL, Composite Process, Network Management, Network Monitoring.

1 Introduction

Integrated management frameworks have traditionally provided a way to use homogeneous procedures to access managed resources. However, the evolution of the networks and the services deployed on them have implied the necessity of new management mechanisms [1]. Currently, new technologies compete in the network management arena, where web services and ontologies can be used respectively for the exchange of management information and the definition of management information itself. Web services provide a maximum decoupling among components and abstraction of the inner complexities with well defined interfaces. Ontologies provide a way to formally describe the management information, avoiding misinterpretations.

Web service composition is another technology with application in network management. A set of web services can be called in a sequence to accomplish the tasks of a management application. The composition of web services can be defined formally by using service ontologies such as OWL-S that describe by a set of processes how and when to invoke these web services. However, current semantic web service tools are not mature enough to interpret such process descriptions. Then, in the meantime, another approach is needed to execute such descriptions in a similar manner, albeit less expressive than a proper ontology-based representation. For instance, Web Services Business Process Execution Language (WSBPEL) definitions can be used instead, as there exist process engines that can interpret this language.

This paper presents an approach to define and execute composite network management processes based on semantic web service technologies. For this purpose, web services and the semantic web technologies are introduced in next section. Then, the representation of composite processes with the OWL-S service ontology is presented, showing a case study for network monitoring. Later on, an approach is proposed to cope with the lack of a semantic web service execution environment by redefining the process with WSBPEL, tackling the translation issues. Finally, some conclusions are given.

2 Web Services and the Semantic Web

This section briefly describes the technologies that support this proposal to execute composite network management processes. For this a short introduction to web services is given, followed by a review of ontology-based technologies and an analysis of the confluence of both areas, in the scope of the so-called semantic web services.

2.1 Web Services in Network Management

A web service, as described in [2], is a software system designed to support the interoperable machine-to-machine interaction through a communication network. To achieve this goal, web services describe their functionality with the Web Service Description Language (WSDL) and they interact with each other by exchanging SOAP messages serialized in XML and sent over a transport protocol, usually HTTP.

The benefits of introducing a web service layer to encapsulate basic functionalities that are useful for network management have already been studied in several works [3, 4, 5], which analyze both service granularity and performance aspects. The last one of these papers points out a fundamental aspect in our study: the benefits of obtaining a common and interoperable interface to access a set of basic functionalities for network management, which can be used to build further, more complex processes.

2.2 Semantic Web and Ontologies in Network Management

The semantic web area [6] comprises a set of technologies to change current web from a network of contents and services interpreted and used by humans to a network in which such contents and services can be exploited by software agents. Among these technologies it is especially relevant in this work the use of ontologies.

An ontology is defined in [7] as “a formal specification of a shared conceptualization”. In practical terms, an ontology is a hierarchy of concepts with attributes and relations that defines a terminology to define in consensus semantic networks of interrelated information units. An ontology provides a vocabulary of classes and relations to describe a domain, stressing knowledge sharing and knowledge representation.

The use of ontologies to represent information related to the network management scope has been addressed to a significant extent in recent research [8, 9, 10, 11, 12, 13]. In the work presented here, the line started in [8] is extended by using a common representation ontology to formalize a set of specifications for network traffic monitoring. In this way, those definitions can be used to obtain a uniform access to a set of basic functions, common to different network management protocols, which will be used as a base to define a set of composite management processes based on these definitions.

2.3 Semantic Web Services in Network Management

Semantic web services are a particularly thriving area within semantic web technologies. Their objective is to provide a set of functionalities that can be understood by software systems to exploit (discovery, composition, invocation) these functionalities in an automatic or semi-automatic manner.

In this way, a set of ontologies have been defined that allows the description of these functionalities to achieve this goal. Among these proposals, the most relevant are OWL-S (OWL Services), WSMO (Web Service Modeling Ontology), and SWSO (Semantic Web Services Ontology) [14]. Although all of them share a similar semantics (they describe in the same terms of inputs, outputs, preconditions and effects the information about a functionality), the tools and methods provided by each representation are not so similar. In this paper OWL-S is used to represent the set of basic functionalities to be later exploited to obtain composite processes based on these functionalities, resuming the work described in [15]. For this, OWL-S process description is used, as detailed later. Other work [16] also proposes OWL-S for the description of network management processes. Using these OWL-S descriptions, for example, a generic management application could manage resources based on Web Services, even if it does not know *a priori* how to do it, which can be very useful in autonomic environments.

Up to this point, semantic descriptions have been introduced, but not how to implement described functionalities. A common practice is to ground the semantic descriptions on web services. Thus, a grounding between the semantic description and the WSDL description of the web service is set up, so that when a semantic web service is used, a traditional web service is finally invoked.

3 Composite Processes Representation

Starting from the perspective described in the previous section, the objective of this work is to illustrate a set of techniques to allow the description of web services related to network management. For this, a set of composite processes relevant for network management is specified. OWL-S is used for the process description, as it is presented in next subsection.

3.1 OWL-S Process Representation

OWL-S [17] allows the representation of a service as a set of interactions with other services. To represent this interaction, the ServiceModel class and its subclass Process have been defined. They are based on existing techniques for workflow and process modeling to describe a service as a process. In this context, two kinds of processes can be distinguished: atomic processes and composite processes.

An atomic process receives an input message and returns an output message. Thus, this type of processes can be executed directly. To make it possible, each AtomicProcess class has a Grounding information associated to it, allowing a client to build and interpret the messages interchanged with the service.

A composite process is expressed as a composition of other processes (atomic or composite). This composition can be expressed by the following control structures: sequence, split, split and join, any-order, choice, if-then-else, iterate, repeat-while, and repeat-until. Other specific characteristic of these processes is the data flow. Whereas in an atomic process inputs are generated by a client and outputs are generated by the process, in a composite process, inputs can come from a client or another process, and outputs can be generated by different processes. OWL-S provides constructs to manage the control structures as well as the information flow in composite processes.

Both atomic and composite processes can have two purposes:

1. Change the environment, represented as preconditions and effects.
2. Process data (transform a certain input into a concrete output), represented as process inputs and outputs.

In OWL-S, preconditions and effects are represented as logic formulas. OWL-S does not define a default language to represent such logic formulas. However, it recommends and provides some facilities to work with the Semantic Web Rule Language (SWRL) [18], and gives a mechanism to represent those formulas in other languages. Service inputs and outputs have to be typed with a class of the related domain ontology.

With these tools, it is possible to achieve the objective of creating a complex and interoperable description, based on less complex services, to represent a composite process, which is useful in the network management scope.

3.2 Case Study: Network Monitoring

To illustrate the concepts described above, a detailed case study is provided. In this case, a network traffic monitoring process has been defined to analyze the network load. This process creates a report about network traffic for those interfaces of a probe that have a load with a value higher than a given threshold. For this, it is necessary to define the following set of elements:

- A domain ontology developed in OWL that represents the network traffic management domain. For this purpose, we have used the work in [9], whereby RMON-MIB (RFC 2819) is translated into OWL as a set of classes and properties.
- A set of web services that encapsulate the functionality provided by the RMON-MIB. One service has been generated automatically for each object group of the MIB, defining configuration functions needed to create, modify and delete monitoring tasks, and information retrieval functions needed to obtain the results of

the monitoring tasks. The semantics of the defined tables has been extracted to distinguish between a configuration table, that includes read-create objects and an EntryStatus (or RowStatus) column, and a results table, which includes read-only objects. Fig 1 shows an example of the operations generated for the tables hostControlTable and hostTable, in pseudo-code, of the RMON-MIB host object group.

- Finally, these web services are used as a grounding for a set of OWL-S descriptions. These descriptions represent the services, and relate them with the concepts contained in the domain ontology defined before. Also, SWRL rules are defined, as described in [11], in order to establish how the represented service interacts with the real world.

```
hostControlIndex createHostControlEntry(
          hostControlDataSource, hostControlOwner)
void removeHostControlEntry(hostControlIndex)
void modifyHostControlDataSource(
          hostControlIndex, hostControlDataSource)
void modifyHostControlOwner(
          hostControlIndex, hostControlOwner)
HostControlEntry[] getAllHostControlEntry()
HostControlEntry getHostControlEntryByHostControlIndex(
            hostControlIndex)
HostControlEntry[] getHostControlEntryByHostControlOwner(
            hostControlOwner)
HostEntry[] getAllHostEntry()
HostEntry[] getHostEntryByHostIndex(hostIndex)
HostEntry[] getHostEntryByHostAddress(hostAddress)
```

Fig. 1. Operations generated for the RMON-MIB host object group

Then, the monitoring process can be described by using these elements. Fig. 2 shows the modeled process. This process takes the following steps:

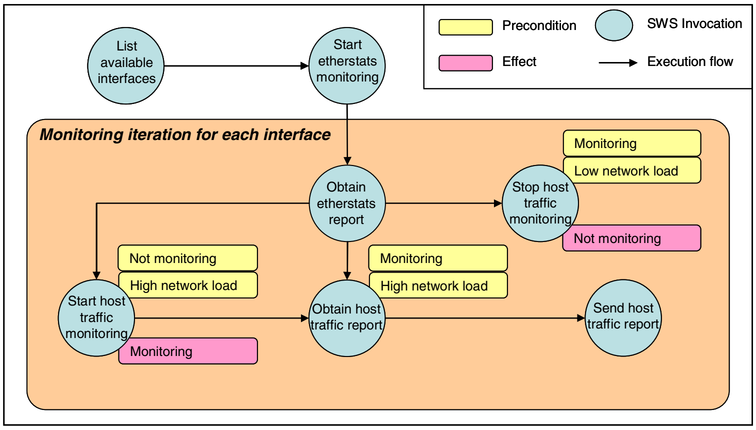


Fig. 2. Conceptual representation of the traffic-monitoring OWL-S process

1. Call the service operation “List available interfaces”, based on IF-MIB (RFC 2863) ifEntry. This service takes a void input, and offers an output with information about all available interfaces in the network probe.
2. Call the service operation “Start etherstats monitoring”, based on RMON-MIB etherStatsEntry. This service takes as an input the interface list to monitor, and starts the monitoring task, obtaining Ethernet statistics for each interface.
3. For each interface:
   a. Call the service operation “Obtain etherstats report”, based also on RMON-MIB etherStatsEntry.
   b. If the preconditions “high network load” and “not monitoring” are met, call the service “Start host traffic monitoring”, based on RMON-MIB hostControlEntry. This service starts the monitoring of each host in a concrete interface of the probe. If it is correctly invoked, call the service operation “Obtain host traffic report”, described below.
   c. If the “high network load” and “monitoring” preconditions are met, call the service operation “Obtain host traffic report”, based on RMON-MIB hostEntry. This service obtains the report of traffic by host in a concrete interface of the probe. If it is correctly invoked, call the service operation “Send host traffic report”, in charge of sending reports to a network manager.
   d. If the “low network load” and “monitoring” preconditions are met, call the service operation “Stop host traffic monitoring”. This service stops the monitoring of hosts in a concrete interface of the probe.

4 Implementation Approach: Use of WSBPEL

Although the formal approach has been introduced, it is necessary to make an extra effort when working with semantic web technologies, because current tools are still under development. Then, first of all, a revision of currently available OWL-S tools has been done. Among them, only Mindswap’s OWL-S API[1] and CMU OWL-S VM[2] provide some support to execute semantic web services from an OWL-S description, although with important limitations. Neither the if-then-else and repeat-while control structures, nor conditional outputs and effects are supported by the OWL-S API, unless custom extensions are introduced. The CMU OWL-S VM is not sufficiently documented to assess the level of support provided by this tool for the execution of complex semantic web service descriptions. Other tools also exist, as stated in [15], but they are just devoted to the edition of OWL-S instances.

Due to these limitations, and given that the defined semantic web services are grounded on a conventional web services, other existing technologies for web service composition have been studied. In this way, if the semantic web services are grounded on a traditional web service, process descriptions can also be grounded on traditional web service composition technologies. In this scope, there are three main approaches: WSBPEL (Web Services Business Process Execution Language), WSCI (Web Services Choreography Interface), and BPML (Business Process Modeling Language).

[1] http://www.mindswap.org/2004/owl-s/api/
[2] http://projects.semwebcentral.org/projects/owl-s-vm/

However, only WSBPEL currently provides a sufficient mature set of tools, including graphical process editors, execution engines, deployed process managers, process debuggers, etc. Moreover, being an OASIS standard, WSBPEL is highly accepted, and has the support of a large community of users.

4.1 WSBPEL Process Representation

WSBPEL [19] defines a model and a grammar to describe the behavior of a business process based on the interactions among the process and its partners. This interaction is achieved by means of web services. Moreover, WSBPEL allows defining how the partners and the process are coordinated to achieve a goal, as well as the state of the interaction and the logic needed to make this coordination possible. Finally, WSBPEL provides a mechanism to describe the way in which some activities have to be compensated or undone if any error occurs in the business process. Then, WSBPEL provides a language to generate process descriptions, independent of the platform, and supporting the definition of all the fundamental aspects of processes.

As it can be observed, a WSBPEL process implementation externally consists of a web service, which defines a set of operations to let other systems interact with the process. Internally, however, a WSBPEL process consists of a complex business process description, which includes variables, partners, error handling and business flow definition.

The variables section is composed of the variable descriptions used by the process, providing its definition in terms of WSDL messages, XSD (XML Schema Data type) types or XML Schema elements. These variables are useful to maintain data and information related to the process status, based on the exchanged messages at a certain time. To access these variables, XPath expressions can be used.

The partners or partnerLinks section describes the behaviour of each web service that interacts with the process. Each partner is defined by a type and a role. This information represents the functionality that a partner has to provide so that the process performs correctly.

The error handling section allows the definition of the actions to be done when an error occurs during the execution of a business process.

The definition of a business flow allows the description of the set of activities to be done in order to achieve the goals defined for the business process. For this purpose, WSBPEL offers a wide set of primitives to deal with data, message reception and transmission, service invocation, conditional expressions and other control structures.

Finally, WSBPEL can be considered a sufficiently expressive language to be used for the execution of the composite processes described in OWL-S. Nevertheless, there are some aspects that WSBPEL cannot cover. The next subsection studies the viability of using WSBPEL to support the execution of OWL-S definitions.

4.2 OWL-S Process Grounding on a WSBPEL Description

The grounding of an OWL-S process on a WSBPEL description is relatively easy to do. WSBPEL offers control structures that are similar to OWL-S structures. At the same time, other functionalities (data flow, variable declaration) are also similar in both descriptions. However, there are some issues to be taken into account: service, data and logic expression descriptions. Then, this subsection analyzes those points in which both technologies differ, which instruments can be used to solve these differences, or what functionality is lost if WSBPEL is used instead of OWL-S. The inverse approach (i.e. a translation from WSBPEL to OWL-S) can be found in [20].

The first aspect to deal with is related to the types used when defining process data. In OWL-S, data are typed by an OWL class or a basic XSD type. However, in both WSDL and WSBPEL descriptions, data are represented by a basic XSD type or a type described with XML Schema. Thus, a translation from an OWL class instance to an XML Schema element is needed. For this kind of translation, document transformation languages such as XSLT (eXtensible Stylesheet Language Transformation) are commonly used. Nevertheless, this process is usually not trivial, because in OWL and other ontology languages based on description logics, classes can be defined as a set of restrictions, and the form of an instance is not easily known. It is worth mentioning that this problem is not common in network management ontologies, because most ontologies are derived from existing MIB or CIM schema specifications, based on objects and properties. Another consideration is about the unique identification of an instance with a URI, which is lost when transforming it to an XML Schema data type. Once again, this problem is not common in network management ontologies, in which functional properties are usually used to identify a concrete instance of a class.

The next aspect is related to logic expressions and their use in both OWL-S and WSBPEL. Logic expressions in OWL-S are mainly used to define conditions in control structures, preconditions and effects. As stated before, WSBPEL allows the use of control structures, but it does not have preconditions and effects when calling a partner. Then, OWL-S preconditions and effects have to be extracted from each service call, and included in the process flow to achieve a functional correspondence in the WSBPEL process. This extraction cannot be easily automated, so it has to be done by hand. Another relevant issue is the expressiveness of the logic expression languages used in OWL-S and WSBPEL. WSBPEL does not provide such a language, using XPath instead. XPath [21] is a language to manipulate XML with a set of added functions, such as arithmetic comparisons (<, >, =) and simple Boolean expressions (and, or, not). On the other hand, OWL-S proposes the use of SWRL to define logic expressions, which joint with the OWL descriptions provides a higher expressiveness than XPath. Given that current WSBPEL engines do not support SWRL, the logic expressions contained in the OWL-S descriptions have to be limited so that they can be translated to XPath. Then, this translation cannot be done automatically.

Finally, the description of partners has to be analyzed. As commented before, WSBPEL allows the definition of roles for those partners involved in the process. This definition is done based on the set of operations that a partner provides. Semantic Web techniques aim at allowing partner descriptions to be presented in terms of what is going to be obtained instead of describing a communication interface. Given this fact, and keeping in mind that the objective of this work is to obtain a practical result, this kind of partner descriptions have to be avoided. Instead, just operations, inputs and outputs, along with the appropriate XML Schema mappings, should be defined.

4.3 Application to the Case Study: Network Monitoring Process in WSBPEL

Once the WSBPEL process representation and its relationship with OWL-S have been described, this subsection explains the adaptation to WSBPEL of the case study presented in subsection 3.2, where an OWL-S specification was defined for a network monitoring process.

First of all, it is necessary to bind all data. For this purpose, a transformation is performed from the OWL class instances, defined as service inputs and outputs, to the XML Schema data types of the web services, which encapsulate the RMON functionality. There are several ways of doing this binding, among which XSLT transformations are our proposed approach in this work.

Next, it is necessary to model the OWL-S composite process in WSBPEL. As mentioned before, this translation is complex and cannot be done automatically, so it has to be done manually, taking advantage of the available editing tools. In our work, the ActiveBPEL Designer[3] editor has been used. The translation process has been as follows:

1. Include in the specification all the web service calls needed to complete the process. During this step, it is necessary to define the partner profiles for the process. That is, the set of methods that any network probe has to implement. Given that semantic web services are used, this definition can be done in terms of objectives (preconditions and effects) instead of inputs and outputs. However, due to the problem mentioned above, this description has to include the required operations, as well as their inputs and outputs, in order to obtain an executable WSBPEL process.
2. Define the flow and control structures needed to execute the process. In this step, the control structures used in the OWL-S description are translated to WSBPEL structures. This process also requires the translation of the logic expressions used in the OWL-S specification to those of the WSBPEL description. This is only possible if the expressivity of OWL-S expressions is limited to fit in the accepted WSBPEL expressions.
3. Extract the logic introduced in the preconditions and effects of the OWL-S description, and integrate it in the WSBPEL process definition. This step has to be done again manually for each precondition and effect.

One important aspect to translate the OWL-S description to WSBPEL is the role that performs the reasoner when processing OWL-S descriptions. In OWL-S processes, the definition of memory structures does not exist, because it is the reasoner who takes care of it. However, when describing a WSBPEL process, it is necessary to specify all the data structures to be used. Then, it is possible that during translation, some auxiliary variables have to be declared, and the management of these variables (access, init values, etc.) needs to be specified. If all these facts are taken into account, the result is a WSBPEL process that can be loaded into a BPEL engine and run as shown in Fig. 3.

In this work, ActiveBPEL Engine[4] has been used to run the WSBPEL process. The result of this development is a WSBPEL process definition that implements the functionality contained in the OWL-S process description. This WSBPEL definition

[3] http://www.active-endpoints.com/freebpel/
[4] http://www.activebpel.org/

presents a WSDL service interface that can be used as a grounding for the OWL-S Service description. Thus, this WSBPEL process definition is completely interoperable, so it can be deployed in any WSBPEL engine. The location of the component services implied in the WSBPEL process description can be modified using a WS-Address. In Fig. 4, the process deployed in the ActiveBPEL Engine is shown.

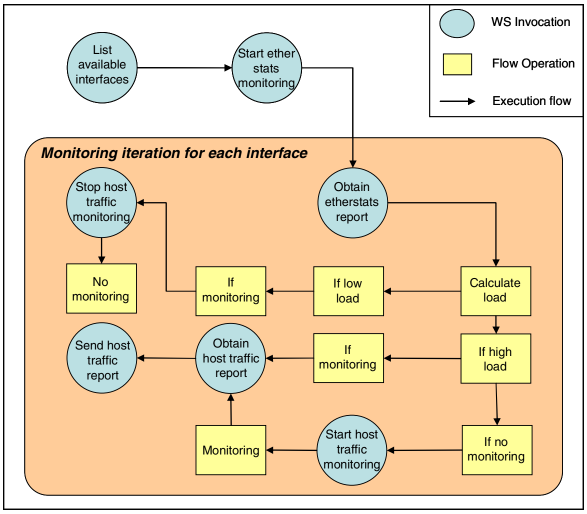
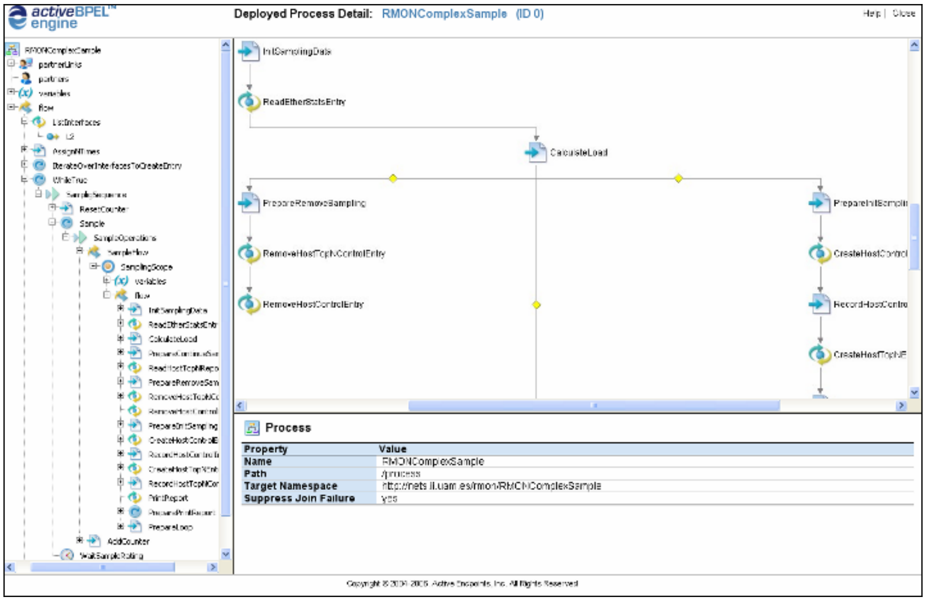


Fig. 3. Conceptual representation of the traffic monitoring WSBPEL process



Property	Value
Name	RMONComplexSample
Path	/process
Target Namespace	http://nets.ii.uam.es/rmon/RMONComplexSample
Suppress Join Failure	yes

Copyright © 2004-2006 Active Endpoints, Inc. All Rights Reserved

Fig. 4. Load-based Network Management process deployed in the ActiveBPEL Engine

5 Conclusions

Web service technology allows the definition of network management interfaces to be deployed on the network resources. These services are usually combined to perform a management task, but WSDL specifications only provide the information related to each interface. To address this problem, service ontologies, such as OWL-S, are useful to define the relation among different web services in a management process. This definition can be interpreted by a manager, which calls the services following a sequence with control structures. The advantage of this approach is the shift of the application development workload to a process definition, aided by graphical editors which directly generate that definition from a flow diagram. This paper has presented a case study in which OWL-S has been used to describe the composite process to monitor the traffic load of a network.

Due to the necessity of using an execution engine to interpret such definitions, this work has also studied how to translate an OWL-S definition to WSBPEL. Thus, until future OWL-S engines make this task unnecessary, the defined composite process has been translated to WSBPEL and loaded into an execution engine, performing the network monitoring previously described. Using currently available WSBPEL engines has several benefits, including the use of BAM (Business Activity Monitoring) technologies [22] to monitor and assess the correctness and quality of the deployed processes. When OWL-S engines are available and related technologies like BAM can work with such engines, a future task shall be to load the defined semantic process and check if they perform as foreseen.

Given this approach, one may think that WSBPEL can be used directly in most of cases to combine web services for a management application. However, WSBPEL is somehow limited, as web services must comply with a set of defined inputs and outputs. On the other hand, the semantics of OWL-S enable the future definition of autonomic systems that can interpret the semantics of the processes to achieve their goals. Future process execution engines will either use OWL-S Process descriptions or should improve current WSBPEL, importing some of OWL-S semantics key points identified in this work.

In our envisioned future work we shall also study the application of these technologies to other management functional areas, following the FCAPS (Fault, Configuration, Accounting, Performance and Security) model, to assess the feasibility of such management architecture.

References

1. J. Schönwälder, A. Pras, J.P. Martin-Flatin: On the Future of Internet Management Technologies. IEEE Communications Magazine, Vol. 41, Issue 10 (2003) 90-97.
2. H. Haas, A. Brown: Web Services Glossary. W3C Working Group Note (11 February 2004)
3. G. Pavlou, P. Flegkas, S. Gouveris, A. Liotta: On Management Technologies and the Potential of Web Services. IEEE Communications Magazine, Vol. 42 Issue 7 (2004) 58-66.
4. A. Pras, T. Drevers, R. van de Meent, D. Cuartel: Comparing the Performance of SNMP and Web Services-Based Management. eTransactions on Network and Service Management, Vol. 1, No. 2 (2004) 72-82.

5. T. Fioreze, L.Z. Granville, M.J. Almeida, L. Tarouco: Comparing Web Services with SNMP in a Management by Delegation Environment. In. Proc. 9th IFIP/IEEE Intl. Symp. on Integrated Network Management (IM 2005), Nice, France, (May 2005) 601-614.
6. T. Berners-Lee, J. Hendler, O. Lassila: The Semantic Web. Scientific American, Vol. 284, No. 5 (2001) 34–43
7. T. R. Gruber: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, Vol. 5, No. 2 (1993) 199-220.
8. J.E. López de Vergara, V.A. Villagrá, J.I. Asensio, J. Berrocal: Ontologies: Giving Semantics to Network Management Models. IEEE Network, Vol. 17, Issue 3, (2003) 15-21.
9. J.E. López de Vergara, V.A. Villagrá, J. Berrocal: Applying the Web Ontology Language to management information definitions. IEEE Communications Magazine, Vol. 42, Issue 7 (2004) 68-74.
10. S. Quirolgico, P. Assis, A. Westerinen, M. Baskey, E. Stokes: Toward a Formal Common Information Model Ontology. WISE'2004, Lecture Notes in Computer Science, Volume 3307, Springer Verlag (2004) 11-21.
11. A. Guerrero, V.A. Villagrá, J.E. López de Vergara, J. Berrocal: Ontology-based integration of management behaviour and information definitions using SWRL and OWL. DSOM'2005, Lecture Notes in Computer Science, Vol. 3775, Springer Verlag (2005) 12-23.
12. A.K.Y. Wong, P. Ray, N. Parameswaran, J. Strassner: Ontology Mapping for the Interoperability Problem in Network Management. IEEE Journal on Selected Areas in Communications, Vol. 23, Issue 10 (2005) 2058-2068.
13. J. Keeney, D. Lewis, D. O'Sullivan, A. Roelens, A. Boran, R. Richardson: Runtime Semantic Interoperability for Gathering Ontology-based Network Context. In Proc. 10th IFIP/IEEE Network Operations and Management Symposium (NOMS'2006), Vancouver, Canada (April 2006)
14. M. Burstein, C. Bussler, T. Finin, M.N. Huhns, M. Paolucci, A.P. Sheth, S. Williams, M. Zaremba: A semantic Web services architecture. IEEE Internet Computing, Vol. 9, Issue 5 (2005) 72-81.
15. J.E. López de Vergara, V. A. Villagrá, J. Berrocal: Application of OWL-S to define management interfaces based on Web Services. MMNS'2005, Lecture Notes in Computer Science, Vol. 3754, Springer Verlag (2005) 242-253.
16. J. Keeney, K. Carey, D. Lewis, D. O'Sullivan, V. Wade: Ontology-based Semantics for Composable Autonomic Elements. In Proc. Workshop on AI in Autonomic Communications at the 19th International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, Scotland. (July 2005)
17. D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara: OWL-S: Semantic Markup for Web Services. W3C Member Submission (November 2004)
18. I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission (May 2004)
19. A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, C.K. Liu, V. Mehta, S. Thatte, P. Yendluri, A. Yiu, A. Alves: Web Services Business Process Execution Language Version 2.0. OASIS Consortium Committee Draft (December 2005)
20. D. J. Mandell, S. A. McIlraith: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. ISWC'2003, Lecture Notes in Computer Science, Vol. 2870, Springer Verlag (2003) 227-241.
21. J. Clark, S. DeRose, eds.: XML Path Language (XPath) Version 1.0. W3C Recommendation (November 1999)
22. Alan Joch: Containing Business Processes. Oracle Magazine (March/April 2005)

Towards a Managed Extensible Control Plane for Knowledge-Based Networking

David Lewis, John Keeney, Declan O'Sullivan, and Song Guo

Knowledge & Data Engineering Group (KDEG)
Centre for Telecommunications Value Chain Research (CTVR)
Department of Computer Science,
Trinity College Dublin,
Dublin, Ireland
{Dave.Lewis, John.Keeney, Declan.OSullivan, gsong}@cs.tcd.ie

Abstract. This paper proposes an open, extensible control plane for a global event service, based on semantically rich messages. This is based on the novel application of control plane separation and semantic-based matching to Content-Based Networks. Here we evaluate the performance issues involved in attempting to perform ontology-based reasoning for content-based routing. This provides us with the motivation to explore peer-clustering techniques to achieve efficient aggregation of semantic queries. The clustering of super-peers using decentralized policy engineering will deliver the incremental deployment of new peer-clustering strategies.

1 Introduction

As networks, distributed computing and pervasive computing converge, an increasing trend can be observed for making elements in a network more autonomous, be it an IP router, a wireless terminal, an application server or a sensor. This involves delegating more decision making capability to the elements and providing them information on their changing operational context, which can be used in making that decision. For example, much effort goes into the means for distributing information between elements when designing routing algorithms for ad hoc networks. From an operations perspective there is a need for such complex networks to become increasingly self-managed, or autonomic. Autonomic systems use knowledge of their operational state and operational context to self-manage, i.e. to self-configure, self-heal, self-optimize and self-protect, by monitoring state and context, planning and adapting. Though the need to self-manage was initially recognized as a challenge in dramatically reducing the operating costs of complex computing systems, increasingly complex networking systems comprising the Internet and telecommunications networks are also seen as needing to self-manage. Clark et al identified the central role of a knowledge-driven approach to support advanced AI techniques for monitoring and analyzing network conditions in order to drive the planning of optimization, protection or corrective strategies [clark].

The major challenges faced in distributing knowledge in converging networks are:

- Heterogeneity of contextual information: convergence means elements increasingly need to gather contextual information from peer elements of dissimilar types;
- Rapid evolution of applications: resulting in uncertainty about the type of contextual information that an element will need to gather in the future;
- Volatile peer set: with wireless and mobile communications playing an increasing role in integration, a given element will need to gather contextual information from a frequently changing set of peers.

This requires an autonomic knowledge delivery service that can inherently scale to the size of the network it supports. Equally, the recent interest in applying autonomic principles to communications networks has emphasized the need for access to network operational knowledge in an ontological form [mulvenna][stevenson].

In this paper we examine a common, scalable solution for gathering distributed context information that can be used by arbitrary peers in a future converged network. To address the volatility of peer sets we adopt the Publish-Subscribe (Pub-Sub) paradigm [meier] for the gathering of context information. The elements requiring contextual information express an interest through a subscription which is matched to messages published by other elements holding that type of information as it changes. Pub-Sub systems are already used for loosely coupled communication in a variety of applications. However, existing Pub-Sub systems require agreements on message types between the developers of publishing and subscribing applications. This places severe restrictions on the heterogeneity and dynamism of the information elements that can be exchanged. One solution to this is a Pub-Sub system that filters events based on matching client subscriptions to message attributes rather than the full message type, a technique known as content based networking. Content-Based Networks (CBN) thus facilitate still looser coupling between producer and consumer applications than Pub-Sub. Several CBN solutions and prototypes exist, e.g. [carzaniga01][segall][pietzuch][chand][strom]. However, widespread CBN deployments have been slow to emerge. This is partly due to the difficulty in reaching a general compromise between the expressiveness of event types and subscription filters and the need both to match these efficiently at CBN nodes and to efficiently maintain forwarding tables by aggregating new subscriptions with any existing ones that cover a superset of matching messages [carzeniga99]. As a result current CBNs only support a very limited range of data types and operators for use in matching consumer subscriptions to message attributes, typically: Strings, Integers, Booleans and associated equality, greater then, less than, and regular expression matches on strings. This falls well short of supporting the heterogeneity and flexibility that elements in a converged network require to gather operational context. Selecting a more expressive language involves a difficult trade-off, since higher level features, e.g. set functions, introduce more complexity into a CBN node, and may only be of use to a subset of applications. We must aim therefore to have a CBN message and subscription language that can be expanded incrementally and autonomously to meet the requirements of emerging application domains without placing unnecessary overheads on the network as a whole.

Increasingly, researchers are turning to the use of ontology-based semantics to address this issue. The standardization of ontology languages by the Semantic Web initiative at the World Wide Web Consortium (W3C) [berners-lees] has spurred an increasing number of researchers to use ontology-based semantics to support interoperability in evolving systems [wang][masuoka][belecheanu]. However, these solutions have either been centralized or based on middleware scaled only to enterprise networks.

Therefore, we assert that the target for CBN expressiveness should be the subject-predicate-object structure of ontological knowledge representations, standardized as the W3C's Resource Description Framework (RDF) [rdf]. Thereby, subscription queries can contain arbitrary logic based on any binary predicate defined for message attributes. A CBN based on such triple-structure messages and corresponding RDF queries is far more flexible, open and reusable to new applications. We call such a semantic-based CBN a *Knowledge-Based Network* (KBN). Previous work by authors detail a number of prototype systems that have been implemented to partially achieve this [lynch05][lynch06][keeney05][keeney06].

The scalability of a KBN to Internet proportions requires a routing control plane that minimizes both the size of routing state held in KBN nodes and the overhead of ontological reasoning in nodes. At the same time this control pane must itself auto-configure in response to topology changes, exhibit robustness to network failures and maximize reachability. The scalability of the routing control plane in the Internet relies, through the use of the Border Gateway Protocol [rekhter], on the natural administrative partitioning of the Internet into Internet Service Provider (ISPs) domains. However, in a KBN addressing schemes play only a small part in how the knowledge is organised and partitioned in the network. The efficient partitioning of routing space must be based on groupings related to the semantics of message contents rather than grouping within the hierarchies of network addresses.

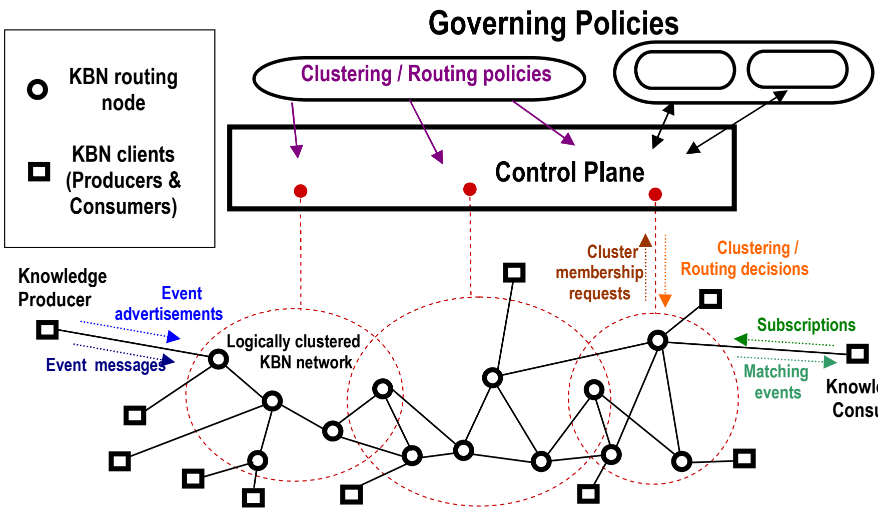

Governing Policies
KBN routing node
KBN clients (Producers & Consumers)
Clustering / Routing policies
Control Plane
Knowledge Producer
Event advertisements
Event messages
Logically clustered KBN network
Cluster membership requests
Clustering / Routing decisions
Subscriptions
Matching events
Knowledge Consumer

Fig. 1. A control plane for a Knowledge Based Network

Clearly, any software-based event forwarding algorithm will struggle to match the hardware optimized performance of packet forwarding in IP routers. Basing the forwarding algorithm on today's ontological reasoners will incur a particularly heavy computational load. However we do not seek to develop optimized reasoners for KBNs. Instead we are exploring the extent to which intelligent clustering in KBN routing algorithms that are cognizant of the performance profiles of existing reasoners and of the semantics being exchanged by client applications, can off-set this relatively poor forwarding algorithm performance.

In the remainder of this paper we propose an architecture for the KBN that separates the routing protocol from the matching and forwarding aspects (Figure 1). This split aims to minimize routing table size, as well as minimizing matching latency due to ontological reasoning. The routing protocols will exploit the robust behaviour and efficient look-up of peer-to-peer networks, and will use hierarchical clustering to minimize the ontological load on individual routers. To correctly dimension such a clustering scheme we must first understand the performance of existing reasoners. To this end we present in section 5 performance measures of such reasoners for use in KBN routers. We also discuss the impact this may have on the design of the routing plane. In addition we discuss related work (section 3) and design issues (section 4).

2 Architectural Principles

To reach its potential in flexibly supporting heterogeneous autonomic systems at Internet scales, the KBN must exhibit the following characteristics:

- Scalability in terms of latency and message throughput by minimizing the reasoning overhead at each router while minimizing routing configuration overhead.
- Robustness to the node and link failures that are inevitable at Internet scales.
- Self-configuration in response to the addition or removal of KBN nodes.
- Governance through policies, so that operational goals and constraints of different administrative domains can be applied in terms of high level rules to the relevant portions of the global KBN.

The use of RDF will support extensibility and applicability to new applications, while allowing the system to be extensible and modular in its routing and self-management functions by allowing co-existence of different routing and management schemes based on different sets of application semantics with configurable reasoning logic. It will support incremental deployment by different administrative domains, initially edge-based overlays, and later in the ISP-operated network core.

We therefore place an emphasis on Internet engineering values, (i.e. incremental deployment of new network features, support for application innovation, minimal standardization of core network features) in addition to the measurable properties of network scalability and performance. Overall, these issues represent a significant departure from the mainstream of knowledge-based systems research that has rarely extended beyond desktop applications and enterprise scope. Though the Semantic Web enables knowledge exchange at a global scale, communication is via HTTP or enterprise-scale middleware infrastructures, and are thereby insulated from the issues of Internet scaling. By addressing asynchronous messaging over a highly decentralized network we uniquely attempt to reconcile Internet engineering values and knowledge engineering solutions, thereby exploiting the new efficiencies yielded by clustering KBN nodes based on semantic concerns.

The significance of the approach taken is that it exploits the ontological knowledge to construct efficient routing schemes through semantic clustering so that routing table sizes can be minimized and the knowledge base at each router can be kept small to minimize reasoning overhead. Clustering thereby both increases the scalability of RDF-based routing and supports the deployment of routing schemes tailored to specific application domains, thus allowing a wide range of strategies to co-exist in rendezvousing events advertisements and subscriptions via the KBN control plane.

3 Related Work

There has been little examination of the use of ontology-based semantics in content-based networking in the scientific literature. In [petrovic], an extension to the Toronto Publish/Subscribe System (ToPSS) is described that proposes extending the event/subscription matching function of this CBN to include class equivalence, ontological sub-class and super-class relationships (i.e. subsumption) and semantic mapping based relationships, which is equivalent the CBN extensions we have implemented as described in [keeney06][lynch05][lynch06]. However, [petrovic] does not address scalability issues of including ontology-based reasoning into the CBN, and no proposal is made to integrate this with the P2P routing extension for ToPSS. More significantly, however, no report of an implementation or evaluation of this proposal has yet emerged. In [li] a semantic publish/subscribe system is presented, but it is based on a centralized pub/sub bus implementation and thus is limited to enterprise scale and does not offer true CBN capabilities.

Much work to date on content-based networks has focused on how efficiency in routing can be gained through subscription aggregation and merging, where routes to subscribing clients are multiplexed with ones with covering subscriptions, i.e. broader subscriptions that will match all the event messages that would match the covered subscription. Recent progress with the XNET CBN has shown that perfect routing can be achieved in a scaleable manner independently of subscriber joins and leave rates though subscription aggregation [chand]. XNET does not, however, address how possible routes from KBN nodes to message producers are established. The default approach is flooding, where a node requests all other nodes for relevant routes, but this is not scaleable to large numbers of nodes [muhl]. This is addressed in the SIENA CBN through the static set up of spanning trees [carzeniga03] from producers to all possible consumers. However, these are then costly to recalculate in the event of configuration change or failures, thus failing our requirements for robustness and self-configuration.

The HERMES CBN [pietzuch], ToPSS [muthasumy] and the REBECCA CBN [tempstra] have all addressed these issues by applying peer to peer distributed hash table (P2P DHT) mechanisms to the formation of routing tables in CBN nodes. P2P DHTs such as CAN, CHORD and Pastry have well known properties of scalability, robustness and self-organization. It should be noted that though P2P system are concerned with efficiently routing queries to matching information sources, they do not address the CBN concern of optimally routing a sequence of asynchronous replies back to the set of querying, or in CBN terms, subscribing clients. P2P DHTs provide efficient routing by using a cost metric keyed to the physical topology of the network resulting in average hop-counts for a route in the order of the log of the number of nodes in the network i.e. O(log(N)). It is the demonstrated strengths of DHT-based routing protocols for CBNs that indicate the appropriateness of peer-to-peer Semantic Overlay Networks as a routing mechanism that meets our requirements for an Internet-scale KBN.

There are several attempts at applying P2P DHT techniques to the retrieval of distributed ontology encoded knowledge information, e.g. in RDF, in semantic overlay networks [tempich][cai][loser]. In supporting an ontology-driven DHT-based P2P routing mechanism for the KBN, the approach outlined in [loser] seems most promising due to its support for peer clustering. Used in this way, peer clustering introduces a hierarchy of peer groups based on policies. Such policies can admit nodes based on semantic closeness, recorded performance, administrative domains or indeed reasoning capabilities. It therefore provides a mechanism for these different routing configuration strategies to co-exist, serving different application domains or user communities in a way that supports incremental deployment and innovation.

The engineering of policies for a potentially complicated network of overlapping clusters, however, manifests as further technical policy engineering challenges. However, we have developed a novel policy engineering platform which uses the concept of communities as a resource and policy grouping abstraction [feeney]. This avoids the need for centralized engineering required in role-based policy schemes, and is therefore well suited to the decentralized evolution of policies by the interconnected network of autonomous decision makers that would characterize the operators of KBN clusters in a global setting. The core technical significance of this architecture is in applying semantic clustering based on reasoned knowledge bases at KBN routers through policy-driven super-peer hierarchies to KBN route management.

4 Design Issues

The fully decentralized approach of peer-to-peer systems results in highly scalable systems that can operate on an Internet-scale, prompting the proposed use of P2P techniques in the KBN control plane. By identifying the more significant concepts used to determine where messages should be routed around the network, it is envisioned that a set of determining keys can be used to semantically cluster peers. By supporting such clustering of producers and consumers, message forwarding can be largely restricted within clusters or between super peers, thereby improving scalability and load balancing. Further, these semantic clusters will be based on application semantics and usage, rather than static address hierarchies, so the management of KBN routing domains will need to respond to the needs and activities of application user communities as much as to ISP administrators, especially when operated as an overlay. As such, application-based grouping semantics will be very heterogeneous and so the control plane in which routing information is calculated and exchanged must be inherently more flexible. It must support the operation of multiple routing domains to reflect application heterogeneity while still supporting the sub-grouping needed to enable efficient dissemination of routing information. In addition, as shown in [carzaniga03], such use of determining keys in the node-level forwarding algorithm of CBNs leads to substantial improvements in performance and scalability. Route recalculation and dynamic clustering of peers will be a relatively expensive operation, and will be effected by the amount of churn in the network as new nodes arrive and depart or new subscriptions and advertisements are published. This will therefore be a key performance indicator for the efficiency and scalability of the proposed scheme. Well optimized routes within and between clusters will still be required to minimize the number of hops required between producers and consumers.

The performance of the dynamic clustering and route determination algorithms will be of importance in terms of the overhead in the network and its ability to stabilize during and after churn. However a key overhead (see next section) is the loading of new ontology classes into a KBN node's reasoner, so the ability of the routing scheme to limit this is a key metric to be observed during simulation. This can be evaluated in terms of how the load of the network is balanced across clusters, but in a manner where traffic can be largely contained within the clusters and short routing paths are maintained between producers and consumers. The performance of the forwarding algorithms will depend on the size and complexity of the routing information stored at each node. This can be evaluated in terms of the average time taken to determine the forwarding strategies for individual messages in terms of the semantics of their content. It will be necessary to evaluate the degree to which new or cancelled subscriptions are either covered by another subscription or cause churn within the network, and contrast this with the heterogeneity of the semantics of the subscriptions present in the network. To further compare performances of the KBN a benchmark is presented in [keeney06b].

5 Initial Evaluation of Knowledge Model Reasoning

Recent exploratory experimentation by the authors has evaluated the integration of ontological equivalence and subsumption into SIENA CBN event/subscription matching [lynch05][lynch06][keeney06] yielding the following results. Firstly, the loading of new ontologies into a reasoner embedded in a KBN node is computationally expensive due to load-time inference, so the frequency of changes to the ontological base of a given KBN node must be minimised since changes will need to be distributed to each of the nodes in the network. Secondly, ontological reasoning is memory intensive and memory usage is proportional to the number of concepts and relationships loaded into the reasoner so reasoning latency can be controlled by limiting this number in any given KBN node. However, once loaded and reasoned over, the querying of such an ontological base is relatively efficient, with performance relative to size of the ontological base. Based on these initial observations, that the (re-)loading and (re-)reasoning of ontologies is expensive and that such operations will greatly affect the scalability of the network, it was considered possible that such axioms could form the basis of semantic clustering policies for partitioning the routing mechanism, thereby localising the effect of such operations and minimising the ontological base at each node and so improving routing performance.

In order to determine the extent to which ontology reasoner implementations differed with respect to initial loading and subsequent querying times, it was decided to undertake an experiment to compare OWL Description Logic (DL) implement-tations. Three typical DL implementations are measured in our experiment: Racer [haarslev], the Jena framework [jena], and Pellet [parsia]. Racer implements a highly optimized tableau calculus for very expressive description logic and offers reasoning service for multiple TBoxes and for multiple ABoxes. Pellet is an open-source Java based OWL DL reasoner and supports the full expressivity of OWL DL including reasoning about nominals (enumerated classes). Two Pellet structural reasoners were tested: one is the "`OWLReasoner`" designed to provide an Jena-like interface to applications using Jena to access ontologies by translating the results of common calls to provide Jena data structures; the second "`OWLAPIReasoner`" reasoner is designed to support applications using the OWL API framework [owlapi] to query and manipulate ontologies. Both Pellet and Racer implement tableaux optimization techniques to improve their description logic reasoning performance. Finally, the Jena framework supports a number of different inbuilt reasoning modes, four of which are discussed here: The first, the "`OWL_MEM_RDFS_INT`" reasoner, incorporated directly into Jena, performs RDFS entailment reasoning. The second "`OWL_MEM_TRANS_INF`" reasoner, also supplied with Jena, performs transitive reasoning. The third reasoner configuration is where Pellet is plugged directly into Jena to perform full OWL DL reasoning and is accessed directly from within Jena like one of Jena's own reasoners. The fourth mode "`OWL_MEM`", is where Jena undertakes no reasoning, and is included for comparison.

A thorough and fair comparison of all aspects of these implementations is difficult, because they are implemented in different programming languages with the reasoning algorithms varying considerably. However for our purposes this is not important, rather we were interested in the performance of the implementations given a certain expected pattern of usage. This pattern of usage for knowledge routing is where the ontology is loaded and reasoned over initially at KBN start up time (which we call the loadtime stage) and then the ontology would be subsequently queried on a repetitive and ongoing basis (which we call the runtime stage) to facilitate the KBN routing. Thus in the experiment, for a number of ontologies, loadtime performance is given as the times taken for the different reasoners to load, parse and check the ontology, combined with the time taken to perform TBox classification, perform ABox realisation and an initial query of all concepts. The time taken to perform subsequent queries for the set of concepts in those ontologies was measured as the runtime performance of the reasoners.

For the purpose of investigating the impact of the complexity of ontologies upon the reasoner implementations, five ontologies were chosen for test. These were chosen in order to reflect a range of ontology complexity, from simple ontologies with instances through to complex ontologies with no instances. First, a large but relatively simple subset of DMTF's CIM (Common Information Model) ontology with 167 classes and 733 individuals was used in the test. This ontology contains the set of CIM concepts required to manage printing devices. (A plug-in for the Protégé editor from the Universidad Politécnica de Madrid (UPM) [lópezdevergara02] was used for the conversion of the CIM object-oriented model of classes and properties to an ontological model.) Secondly, we tested a more extensive ontology that represents the entire DMTF CIM Core Model, with 1215 classes and 5734 individuals. This ontology is very large by comparison to most ontologies, but relatively simple in terms of content. Next, the wine ontology [wine] from the W3C community contains a classification of wines, with only 138 classes of which 61 are imported from a food ontology, and 206 individuals of which 45 are also imported. The nominals (individuals in class expression) and advanced DL constructors (e.g. disjunctions and equality) used in wine ontology makes it very complex to reason over, and is used as the de-facto stress test for OWL DL tools. The fourth test data set is the Galen ontology [galen]. It is a medical terminology ontology with a very large and complex class and property model (2749 classes) but without instances. It has traditionally been used as a benchmark for terminological reasoning. Finally, the Mindswap ontology [mindswap] is a relatively small but detailed ontology containing the student and staff details and listings of the Semantic Web Research Group in the University of Maryland. It contains 49 classes, of which 37 are imported, and 122 individuals, of which only 6 are imported.

All tests were performed on a desktop computer with Dual 3.2 GHz Intel Xeon processors, 2GB of RAM, running Windows XP Service Pack 2. For Java-based tools, Sun's JDK 1.5.0 was used. All the timings presented in this section are computed as the average of numerous independent timings.

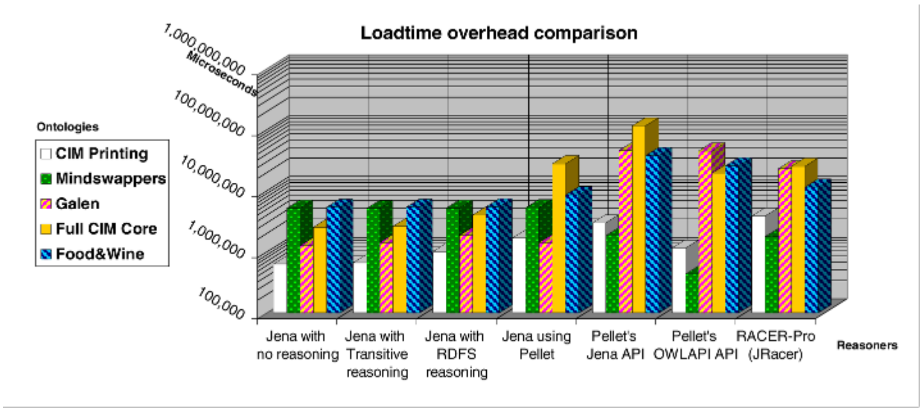


Fig. 2. Loadtime performance of reasoners

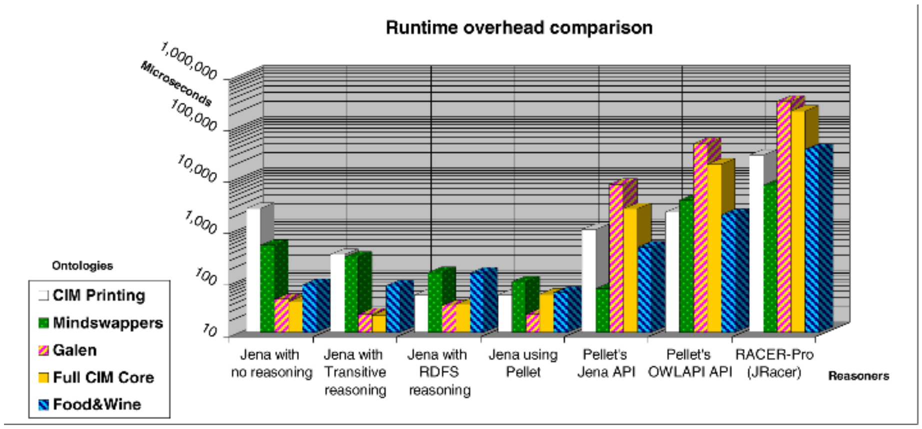


Fig. 3. Runtime performance of reasoners

As can be seen from the loadtime performance (Figure 2) of the different reasoners, all of the reasoners perform similarly with the small ontologies, with the ones accessed through Jena performing a little worse than Racer and Pellet when accessed directly. However for the large and complex ontologies the Jena based reasoners perform substantially better than the others. When the operation of the other Jena based reasoners is compared to the Jena configuration performing no reasoning it is clear that much of the Jena loadtime overhead is independent of the level of reasoning required. Of all of the reasoners, only Pellet, whether or not through Jena, performed an adequate level of reasoning on the complex Wine ontology. For this reason, of the reasoners tested, the Pellet reasoner configurations seems to perform best, perhaps using its own Jena style interface or its OWL API interface for small ontologies, but definitely embedded in Jena for large and complex ontologies.

From the runtime performance graph (Figure 3) it is clear that all of the reasoners accessed using the Jena framework perform far superior to the others. This seems to be achieved by the clever use of model caching for second and subsequent queries within Jena. When compared to the other reasoner configurations, the use of Pellet through the Jena framework would appear to be the best overall choice for runtime performance. However, further investigation may be required to explain the counter-intuitive poor performance of Jena's own reasoners with the smaller ontologies.

In the event that an off-the-shelf generic reasoner is required, taken together, the results point to the adoption of the Pellet reasoner accessed through the Jena framework as the default reasoning configuration. However, in situations where very good runtime performance is required, such as in network routing, and this performance requirement outweighs the need for complete and correct reasoning, the use of cut-down or application specific reasoners may prove beneficial. It is also envisioned that in a large scale network different reasoners will be used by different nodes in accordance with their role in the clustered network's routing scheme.

6 Further Work and Conclusions

In this paper we propose an approach to managing the distribution of routing information between clusters of peers using super-peer communication in order to minimize the impact of ontology-based reasoning in the matching function of a Knowledge Based Network router. We propose that the policies for managing such clustering must therefore be based on the closeness of concepts in a web of ontologies and the likelihood of semantic similarity between subscriptions and advertisement. This work builds on experience gained in extending the basic SIENA CBN router with RDF sub-class and super-class operators, described in detail in [keeney06][lynch05][lynch06]. We do not aim to build a specialized RDF reasoner for this purpose, but to select from the various ones available. Thus in this paper we present the results from comparing several reasoners in order to both select an initial candidate and to understand further the performance trade offs between the addition of new concepts to a reasoners loaded knowledge-based and matching involving those concepts. This latter understanding may be particularly important if semantic clustering prompts certain nodes to specialize in certain groups of types and predicates by integrating reasoners optimized for those semantics and resulting in faster messages matching. This work will continue by evaluating further extensions to the KBN router incorporating the selected Pellet reasoner, both through simulation and through a limited testbed.

Future work will involve integrating an existing community based policy management system [feeney] with the KBN control plane as the mechanism by which clustering policies are defined and potential conflicts identified and resolved. Policy-driven clustering enables the size of the super-peer network and the size and granularity of peer clusters to reflect different application domains needs. This will support overlapping clusters and hierarchies of clusters under separate administrative control. Different policy combinations will be applied to the testbed implementation of the KBN to verify stable behaviour, before those policies are included in the simulated KBN to evaluate the scalability of the resulting clusters. For example, the clustering policy may be specified in terms of accuracy and latency as well as the semantic spread of the query-able knowledge-base, or in terms of queries across a peer population and of the querying load across that population. Though the practicalities and performance impact of policy enforcement will be verified in the testbed, the validation of policy sets and the communication between super-peers in different cluster will be assessed through simulation.

The scalability and flexibility of the KBN under high load of heterogeneity will be evaluated using an OPNET based simulation, in order to test the message overhead involved in P2P clustering and the effectiveness of semantic load sharing. The key semantic distance measures used in clustering will be recorded for each node such that an analysis of the entire network of nodes will be possible in order to evaluate the efficiency of the KBN node clustering scheme that was simulated. A range of semantic distance measures have been proposed, such as [rada][jiang] [sussna][kashyap], and initial investigation will be required to determine the most suitable ones for use in different sizes of clusters or in ontology profiles associated with application domain clustering. The clustering schemes will be initially optimized to cope with a Zipf-like distribution of ontological terms, given that recent analysis of the usage of ontological terms in semantic web documents have shown Zipf-like distribution characteristics [swoogle].

Ultimately we aim to design and validate differing clustering policies that can be used to tune and compliment semantic distance calculations. We will also assess the impact of policies on the coexistence of different (simulated) reasoning capabilities in KBN nodes. It is also planned to extend our work on incorporating semantic interoperability in node matching functions [keeney05][lewis] and in inter-cluster communications.

Further work will also focus on the use of further clustering policies. Such policies can admit nodes based not only on semantic closeness, but also in terms of recorded performance, availability of resources, trustability of nodes, reasoners used and reasoning capabilities, or administrative domains. It therefore provides a mechanism for these different routing configuration strategies to co-exist, serving different application domains or user communities in a way that supports incremental deployment and innovation. This use of clustering policies also supports innovation in clustering strategies by allowing peers to introduce new policy elements and the supporting super-peer matching capabilities.

Acknowledgements

This research was partly supported by Science Foundation Ireland under grant no. 03/CE3/I405, and by the Irish Higher Education Authority under the M-Zones programme.

References

[belecheanu] Belecheanu, R., Jawaheer, G., Hoskins, A., McCann, J.A., Payne, T., "Semantic web meetings autonomic ubicomp" in proc. of the Workshop on Semantic Web Technology for Mobile and Ubiquitous Applications, Hiroshima, Japan, 2004

[berners-lee] Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001

[cai] Cai, M., Frank, M., "RDFPeers: A scalable distributed RDF repository based on a structured peer-to-peer network", in proc of WWW 2004, May 2004, New York, USA.

[carzaniga99] Carzaniga, A., Rosenblum, D., Wolf, A.L., "Challenges for Distributed Event Services: Scaleability vs. Expressiveness" in proc of Engineering Distributed Objects (EDO '99), ICSE 99 Workshop, Los Angeles CA. May, 1999.

[carzaniga01] Carzaniga, A., Rosenblum, D. S., and Wolf, A. L., "The Design and Evaluation of a Wide-Area Event Notification Service", ACM Transactions on Computer Systems, Vol. 19, Issue 3, August 2001

[carzaniga03] Carzaniga, A., Wolf, A. L., "Forwarding in a Content-Based Network" in proc SIGCOMM'03, August 25-29 2003, Kaelsruhe, Germany, ACM Press

[chand] Chand, R., Felber, P.A., "A Scalable Protocol for Content-Based Routing in Overlay Networks", IEEE International Symposium on Network Computing and Applications, April 2003, Cambridge, MA

[clark] Clark, D., Partridge. C., Ramming, J.C., Wroclawski, J.T. "A Knowledge Plane for the Internet", in proc of SIGCOMM'03, 25-29 August 2003, Karlsruhe, Germany

[feeney] Feeney, K. Quinn, K. Lewis, D. O'Sullivan, D. Wade, V., "Relationship-Driven Policy Engineering for Autonomic Organisations", in proc 6th IEEE Int'l Workshop on Policies for Distributed Systems, Stockholm, Sweden, 6-8 June 2005, pp 89-98

[galen] The Galen Ontology, http://www.cs.man.ac.uk/~horrocks/OWL/Ontologies/galen.owl

[haarslev] Haarslev, V., Moller, R. 2001. "RACER System Description", In proc IJCAR 2001, volume 2083 of LNAI, 701–706. Siena, Italy, Springer.

[jena] Carroll, J., Dickinson, I., Dollin, C., "Jena: Implementing the Semantic Web Recommendations", in proc of WWW 2004, New York. May 2004. http://jena.sourceforge.net/

[jiang] Jiang J. Conrath D., "Semantic Similarity based on corpus statistics and lexical taxonomy", in proc of Intl Conference on Research in Computational Linguistics, 1997.

[kashyap] Presentation, NIST Invitational Workshop on Semantic Distance, Gaithersburg, MD, November 2003

[keeney05] Keeney, J., Lewis, D., O'Sullivan, D., Roelens, A., Boran, A., Richardson, R., "Runtime Semantic Interoperability for Gathering Ontology-based Network Context", in proc of NOMS 2006, Vancouver, Canada. 3-7 April 2006

[keeney06] Keeney, J., Lynch, D., Lewis, D., O'Sullivan, D., "On the Role of Ontological Semantics in Routing Contextual Knowledge in Highly Distributed Autonomic Systems", Tech. Report (TCD-CS-2006-15), Dept of Computer Science, Trinity College Dublin. 2006.

[keeney06b] Keeney, J., Lewis, D., O'Sullivan, D., "Benchmarking Knowledge-based Context Delivery Systems", in proc of ICAS 06, Silicon Valley, USA, July 19-21, 2006.

[lewis] Lewis, D., O'Sullivan, D., Power, R., Keeney, J., "Semantic Interoperability for an Autonomic Knowledge Delivery Service", in proc of WAC 2005, Athens Greece, Oct 2005
[li] Li, H., Jiang, G., “Semantic Message Oriented Middleware for Publish/Subscribe Networks”, in proc of SPIE, Volume 5403, pp 124-133, 2004
[loser] Loser, A., Naumann, F., Siberski, W., Nejdl, W., Thaden, U., “Semantic overlay clusters within super-peer networks”, in proc Int’l Workshop on Databases, Information Systems and Peer-to-Peer Computing in Conjunction with the VLDB 2003
[lópezdevergara02] López de Vergara, J.E., Villagrá, V.A., Berrocal, J, “Semantic Management: advantages of using an ontology-based management information meta-model”, in proc of HP-OVUA'2002, distributed videoconference, 11-13 June 2002
[lynch05] Lynch, D., “A Proactive approach to Semantically Oriented Service Discovery”, MSc dissertation, Dept of Computer Science, Trinity College Dublin, 2005
[lynch06] Lynch, D., Keeney, J., Lewis, D., O'Sullivan, D., "A Proactive approach to Semantically Oriented Service Discovery", in proc of IWI’06, at WWW’06, Scotland. 2006.
[masuoka] Masuoka, R., Labrou, Yannis, Parsia, B., Sirin, E., “Ontology-Enabled Pervasive Computing Applications”, IEEE Intelligent Systems, Sep-Oct 2004, pp 68-72
[meier] Meier, R., Cahill, V., “Taxonomy of Distributed Event-Based Programming Systems“, The Computer Journal, vol 48, no 5, pp 602-626, 2005
[mindswap] The Mindswappers Ontology, http://www.mindswap.org/2004/owl/mindswappers
[muhl] Muhl, G., Fiege, L., Gartner, F., Buchman, A., “Evaluating Advanced Routing Alogorithms for Content-Based Publish/Subscribe Systems”, in proc of MASCOT 2002
[muthusamy] Muthusamy, V., Jacobsen, H.A., “Small–scale peer-to-peer publish/subscribe” in proc Workshop on Peer-to-Peer Knowledge Management, San Diego, USA, July 2005
[mulvenna] Mulvenna, M., Zambonelli, F., “Knowledge Networks: the nervous system of an autonomic communication infrastructure”, in proc of WAC 2005, Athens Greece, Oct 2005
[owlapi] Sean Bechhofer, Phillip Lord, Raphael Volz. Cooking the Semantic Web with the OWL API. In proc of ISWC2003, Sanibel Island, Florida, October 2003.
[parsia] Parsia, B., Sirin, E. 2004., “Pellet: An OWL-DL Reasoner”, Poster at ISWC 2004, Hiroshima, Japan, 2004.
[petrovic] Petrovic, M., Burceaa, I., Jacobsen, H.A. “S-ToPSS – a semantic publish/subscribe system” in proc VLDB, Berlin, Germany, September 2003
[pietzuch] Pietzuch, P., Bacon, J., "Peer-to-Peer Overlay Broker Networks in an Event-Based Middleware". in proc DEBS'03 at ACM SIGMOD/PODS Conference. California, June 2003
[rada] Rada R., Mili H., Bicknell E., Blettner M., “Development and application of a metric on semantic nets”, IEEE Transactions on Systems, Man, and Cybernetics 19, pp 17-30, 1989.
[rdf] W3C (1999) World Wide Web Consortium, Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation, http://www.w3c.org/
[rekhter] Rekhter, Y., Li, T., (eds) “A Border Gateway Protocol 4 (BGP-4)”, IETF RFC 1771, March 1995
[segall] Segall, B. et al, “Content-Based Routing in Elvin4”, In proc AUUG2K, Canberra 2000.
[stevenson] Stevenson, G., Nixon, P., Dobson, S, “Towards reliable wide-area infrastructure for context-based self-management of communications”, in proc of WAC 2005, Athens, 2005
[strom] Strom et al., “Gryphon: An Information Flow Based Approach to Message Brokering“, In Intl. Symp. on Software Reliability Engineering 1998
[sussna] Sussna M., “Word sense disambiguation for free-text indexing using a massive semantic network”, proc of Conference on Information and Knowledge Management, 1993.
[swoogle] “Swoogle statistics: Figure 6. Cumulative Term/Namespace usage distribution”, http://swoogle.umbc.edu/modules.php?name=Swoogle_Statistics&file=figure&figurename=figure6

[tempich] Tempich, C., Staab, S., Wranik, A., "REMINDIN': semantic query routing in peer-to-peer networks based on social metaphors" WWW 2004, New York, USA, 2004.

[terpstra] Terpstra, W.W., Behnel, S., Fiege, L., Zeidler, A., Buchmann, A.P., "A peer-to-peer approach to content-based publish/subscribe", in proc of DEBS 2003, ACM Press 2003

[wang] wang, X., Dong, J.S., Chin, C.Y., Hettiarachchi, R., Zhang, Daqing, d., "Semantic Space: An Infrastructure for Smart Spaces", IEEE Pervasive Computing Magazine, Jul-Sep 2004, pp 32-39

[wine] W3C: The Wine Ontology, http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine

Voice Quality on the Internet in 2005 as Measured by www.TestYourVoIP.com

Mark Sylor, Nagarjuna Venna, and Harrison Ripps

Brix Networks, 285 Mill Rd, Chelmsford, MA, 01824
{msylor, nvenna, hripps}@brixnet.com

Abstract. How well can the Internet serve as a network carrying voice calls? To answer that question, we have been running a web site called TestYourVoIP where users can run tests of VoIP Quality from their PC. In this paper we describe how TestYourVoIP operates, the kinds of tests it performs, and the results we have found from those tests. Our findings indicate that while many users will get acceptable call quality, too many will not. We offer some insights as to the reasons for poor quality based on these real world experiences. Lastly, we note that over 2005, voice quality on the Internet has gotten worse.

1 Introduction

All communications, including voice and video, are converging on IP networking technology. IMS (IP Multimedia Subsystem) is but the latest trend towards converged services. While private IP networks run by service providers and large enterprises will be important carriers of converged services, the Internet will remain as the core network to which everyone connects.

While the Internet is ubiquitous, it was designed to carry best-effort data services such as email and the web. Many have long feared that the Internet was not suitable for carrying real time services such as voice calls. That fear raises the question at the heart of this work, “How well can the Internet serve as a network carrying voice calls?” This question is critical for VoIP Service Providers who route calls over the Internet as well as for network providers rolling out next generation services over IP such as IPTV, Metro-Ethernet, and VoIP peering services. After all, some fraction of the calls will traverse the Internet even in walled garden networks.

To answer the Internet Voice Quality question, we have been running a public web site at http://www.TestYourVoIP.com where users have been testing Internet Voice Quality since mid-2004. In that time users of TestYourVoIP have successfully run over 500,000 tests of Internet VoIP Quality. The software and hardware used in TestYourVoIP are based on a Brix Networks product called BrixCare, which is normally used by VoIP service providers in their customer care departments to provide “self help” testing.

This paper describes the results of tests done in 2005, and describes some conclusions we have drawn from those tests.

2 Test Description

TestYourVoIP tests voice quality on the public Internet using hardware and software verifiers located worldwide. Each test run is performed between a software verifier run by a user and one of 7 hardware verifiers located in various cities around the world. Anyone with a web browser and access to the Internet can run a test. Users simply connect their web browser to http://www.TestYourVoIP.com, select a few test parameters and then run the test. The verifiers measure call signaling quality, media quality, and network quality. When the test is completed, the results are displayed to the user, and recorded in a database.

When a user runs a test the user's browser downloads a web page that contains a software verifier implemented as a Java Applet that runs in the user's web browser. The applet is signed by Brix Networks and the user must grant it the privilege to communicate with the hardware verifier.

2.1 Call Signaling Testing

The applet verifier calls the hardware verifier, using SIP as shown in Fig 1. This sequence simulates a call established directly between a calling party and a called party.

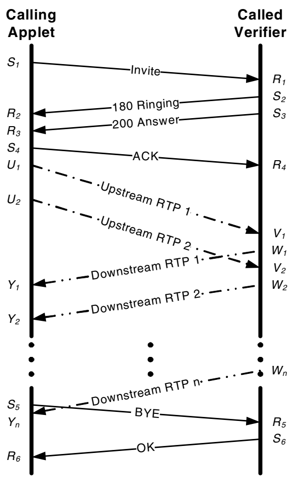


Fig. 1. SIP Call Messages

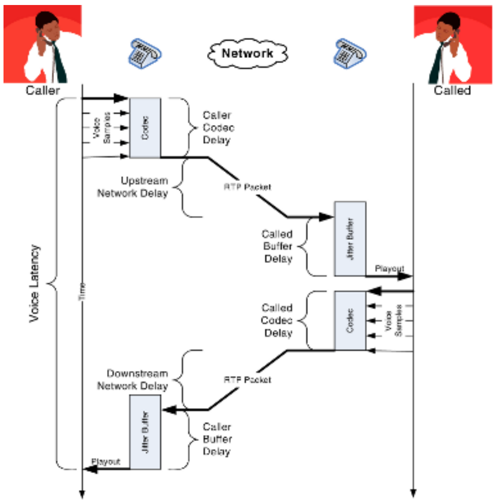


Fig. 2. Voice Latency

2.2 Voice Media Testing

Once the call has been set up, the verifiers exchange voice media traffic in RTP streams over UDP/IP packets. The voice signal is encoded with either the G.711 or the G.729 codec. The specific parameters used in the codecs are listed in Table 1. No silence is inserted in the signal. The call is maintained for 15 seconds. At the end of that time, each party disconnects the call; generally the applet verifier finishes first.

Table 1. Codec Parameters

G.711	G.729
RTP packet sent every 20 milliseconds	RTP packet sent every 20 milliseconds
160 Samples per RTP packet	2 10ms frames per RTP packet
64 Kbits/sec generated by the codec	8 Kbits/sec generated by the codec
80 Kbits/sec generated by IP	24 Kbits/sec generated by IP
87.2 Kbits/sec over the Ethernet link	31.2 Kbits/sec over the Ethernet link

During the call, we count and measure the quality of the received RTP streams. Numerous measurements are made on the RTP traffic; a few of the more important metrics are listed and described in Table 2. Each measurement is made on both streams. In the results that follow, we call the stream sent from the calling applet verifier to the called hardware verifier the upstream media stream and the stream from the called hardware verifier to the calling applet verifier the downstream media stream.

Table 2. RTP Media Stream Metrics

Metric	Description
CMOS	Conversational Quality Mean Opinion Score, an objective measure of voice quality in a conversation.
LMOS	Listening Quality Mean Opinion Score, an objective measure of voice quality when the user is just listening.
Voice Latency	The round trip time of a voice signal from caller to called party and back.
Lost Packets	The percentage of RTP packets not delivered by the network.
Late Packet Discards	The percentage of RTP packets delivered, but too late for them to be played out.
Out Of Order Packets	The percentage of RTP packets delivered out of order.
Duplicate Packets	The percentage of RTP packets received as duplicates of packets already received.
Average Jitter	A measure of variation in network delay in delivering packets.

Of these metrics, the most important is the objective measurement of the Mean Opinion Score (MOS) for voice quality. MOS takes into account multiple sources of degradation in voice quality. MOS is determined using the methodology described in the G.107 standard.

We also send RTCP packets between the verifiers. RTCP allows us to measure voice round trip latency. The user applet verifier may fail to establish the SIP call with the hardware verifier in some cases. One common cause is the presence of devices on the network that block traffic going out on the standard UDP port used by SIP (5060). If a test fails because of call setup problems, we run it again using an alternate port. In general, media is sent and received on a randomly chosen even numbered port between 50000 and 51000. The RTCP stream is sent and received on the next odd numbered port. The tests send the media streams as normal, best effort IP traffic. No special QOS marking is done to the packets; neither in the Ethernet LAN header nor in the IP packet header.

After the media test completes, the call is disconnected using SIP. Results are reported back to the TestYourVoIP site by both verifiers, and the test results are displayed to the end user.

2.3 MOS

MOS (Mean Opinion Score) is the most commonly used measure of Voice Quality. What follows is a summary of the techniques used to derive MOS. For a more complete and detailed explanation, see [1,2].

MOS scores are based on subjective tests of voice quality done by representative telephony users in a lab environment. A group of users are asked to listen to a recorded voice sample through the system under test and score the quality of the voice call on a scale from 1 to 5 where 1 is Bad (or Unacceptable), 2 is Poor, 3 Fair, 4 Good, and 5 Excellent. The individual scores are averaged to derive the Mean Opinion Score.

The user's perception of voice quality depends on the task being performed. Simply listening to a voice sample for understandability and legibility is less demanding than attempting to hold a conversation over a network. Thus two types of MOS scores have been developed, Listening MOS and Conversational MOS.

Subjective testing is difficult and expensive to operate. It also is limited to laboratory test environments. So a number of objective techniques for measuring voice quality have been developed over time. They all take measurements of voice calls, either test calls, or observed calls, and compute an estimate of the MOS score that would have been given to that call by users. The technique used in TestYourVoIP is based on the E Model standardized in ITU G.107. The E Model is packet based, in that it measures statistics such as packet loss and delay and uses those metrics to estimate CMOS and LMOS in an objective fashion. The E Model is efficient to measure and compute, thus making it a practical for large scale testing.

The E Model computes a value called the R Factor, a transmission quality rating, based on the assumption that the causes of degradation are additive. The formula is

$$R = R_0 - I_s - I_d - I_e + A \tag{1}$$

Where R_0 is a base factor that depends on noise and loudness, I_s represents signal impairments, I_d represents impairments that are delayed with respect to speech like echo, I_e represents equipment impairments, and A represents an advantage factor, the users willingness to tolerate poorer voice quality in exchange for convenience. These impairments are derived from: the codec, audio signal loss due to lost packets, audio signal loss due to late packet discards, and voice round trip latency (delay). For details on how the impairment factors are computed, see [1].

The R Factor is then converted into a MOS score. The relationship between R Factor, MOS, and user satisfaction is given in Table 3. In this paper, we use the average of upstream and downstream CMOS scores as the CMOS score for each test.

Too put this value into perspective, a MOS score of 4.4 is generally considered to be toll grade quality. The maximum MOS that can be achieved with the G.711 codec is 4.4. While there is no universal agreement on what constitutes acceptable call quality for VoIP, we count tests that achieve an average CMOS score of 3.6 or better as having acceptable call quality.

Table 3. R Factor to CMOS Conversion

R Factor	CMOS	User Opinion	Users scoring call Good or Excellent
90 – 100	4.3 – 5.0	Very Satisfied	97%
80 – 90	4.0 – 4.3	Satisfied	90%
70 – 80	3.6 – 4.0	Some Users Dissatisfied	70%
60 – 70	3.1 – 3.6	Many Users Dissatisfied	50%
50 – 60	2.6 – 3.1	Almost All Users Dissatisfied	20%
0 – 50	1.0 – 2.6	Not Recommended	0%

3 Related Work

Numerous studies of Internet performance have been made, but we are unaware of any studies similar to the study done at TestYourVoIP. A recent survey of publicly available measurements can be found in [5]. One source of such measurements can be found at CAIDA[6]. Most Internet performance studies have set up a mesh of active testers. Examples include AMP and NLANR. These studies are based on a limited number of testers located in universities and other network hosting sites, and so does not have the scope of TestYourVoIP. Two research efforts, DIMES [8] and NETI@home[7], are attempting to build a larger mesh of testers by asking users to download software that runs in the background following the SETI@home model. DIMES is collecting data on Internet topology through traceroute data. NETI@home is focused more on performance data collection for tests between sites. None of these studies are specifically designed to measure VoIP quality.

A number of publicly available sites on the Internet allow users to run speed tests, often to assess if a user can expect acceptable voice quality. These tests measure available bandwidth upstream and downstream using a variety of techniques. They do not take into account the effects of loss, discards, or voice delay, nor do these sites compute MOS. None of the sites we are aware of have published results.

Keynote Systems conducted one test that does focus on VoIP Quality over the Internet[10]. This test was run between San Francisco and New York over a number of Internet service providers and VoIP providers. The purpose of the test was to compare the quality provided by those ISPs. It did not cover the scale (in either space or time) covered by this study.

The goals of the study reported on in Markopoulous, et.al.[4], are similar to those of our work. However, they are focusing on the Internet Backbone between 5 cities and thus do not include the user access network. Because the user access link is likely to be the main source of loss and delay variation, we believe our study is more reflective of actual user experience. Similarly, the study reported on by Jiang, et.al.

[9] looks at the availability of VoIP on the Internet. It too uses a mesh of active tests between 14 sites, half of which are Internet2 sites and half of which are connected to the Internet via commercial ISPs via Cable Modem or ADSL links. It does not cover voice quality once a call has been made, nor does it cover the scope of this study.

4 Results

In this paper we report on the following questions.

1. What voice quality can users of VoIP over the Internet expect?
2. What are the causes of degradation?
3. What is the affect of codec choice on voice quality?
4. What is the trend in voice quality over time?

The body of data we have collected offers a wealth of information on other questions as well. See the section on Future Work below for some of the other questions we are exploring.

4.1 Filtering Out Bad Network Connections

One reason users run a test on TestYourVoIP is to see whether their Internet connection is good enough to use for VoIP calls. For some tests, the user's Internet connection is not good enough. Tests from bad connections skew the results, and so we have filtered them out of the results we report here. The criterion used to filter the tests was to exclude tests where the round trip voice latency was greater than 1 second or more than 20% of the packets were lost or late packet discards in either direction. Note that in these tests, the MOS score is usually very low (in the Unacceptable range). 9.1% of the tests were filtered out by this criterion.

4.2 Basic Results and ACQ

During 2005, the Average Conversational Objective MOS score across all the included tests run using TestYourVoIP was 3.94. This is a good score, which while not toll quality would satisfy users.

Averages can hide important details. While the users may be happy with the average call, a few bad calls can outweigh the average experience. The distribution of CMOS scores thus gives a better picture of user voice quality experience than the average. Fig 3. shows a cumulative probability distribution of the CMOS scores seen from best (5.0) to worst (1.0). Table 4 below lists the percentage of calls that fall into the quality ranges listed in Table 3.

Most users are satisfied with a call where the CMOS is 3.6 or better. The percentage of test calls that achieve a CMOS score of 3.6 or better is the Acceptable Call Quality Percentage (or $ACQ_{3.6}$). The tests show that during 2005 the tests run on TestYourVoIP achieved an $ACQ_{3.6}$ of 81.1%. While commendable, there remain 18.9% of tests whose CMOS fell below 3.6. Or to put it another way, about 1 out of 5 test calls did not have acceptable call quality. This voice quality is unlikely to satisfy most users of VoIP on the Internet. An $ACQ_{3.6}$ of 95% or 99% would be closer to the quality that users would expect for Telephony over the Internet.

Table 4. CMOS Cummulative Probability Distribution (CPD) by Range

CMOS	Density	CPD
4.3 – 5.0	31.1%	31.1%
4.0 – 4.3	31.9%	63.0%
3.6 – 4.0	18.1%	81.1%
3.1 – 3.6	9.2%	90.3%
2.6 – 3.1	5.1%	95.4%
1.0 – 2.6	4.6%	100.0%

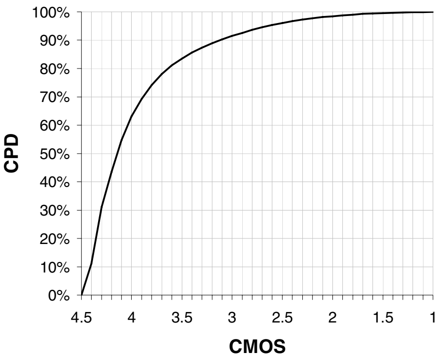


Fig. 3. CMOS Cumulative Distribution

Table 5. Degradation Factors for all calls

Cause	Absolute	Percentage
Codec	0.75	70.0%
Late Discards	0.17	16.3%
Delay	0.09	8.5%
Lost	0.06	5.2%

Table 6. Degradation Factors for poor quality calls

Cause	Absolute	Percentage
Codec	0.77	37.4%
Late Discards	0.49	23.6%
Delay	0.46	22.4%
Lost	0.33	15.8%

4.3 CMOS Degradation Factors

A number of factors affect the computed CMOS score using the E Model. They are: the codec used, lost packets, late packet discards, and the round trip voice latency (delay). The codec degradation is fixed by choice of codec; the others vary with network effects. Because these impairments are additive in the E Model computation of the R Factor, we can compute the relative importance of the degradation causes by the percentage of degradation they cause. By comparing the degradation factors, we can identify the largest contributor to a poor MOS score, the one that should be improved first.

Table 5 shows the CMOS degradation factors that determine the Average CMOS for 2005 as 3.94 both in absolute terms (CMOS points degraded) and as a percentage. The largest factor is the codec, followed by discarded packets, delay and lost packets.

Table 6 shows the degradation factors for tests with low voice quality (CMOS < 3.6). Each factor has roughly the same magnitude, implying there is no single answer to improving those calls with unacceptable voice quality. Each degradation cause must be addressed if we hope to raise the $ACQ_{3.6}$ of the network.

Lost and Discarded Packets. The effect of a lost packet and a packet that arrives too late to be played out is the same, lost audio and degraded call quality. However the cause of lost packets and late packets is quite different. A packet is lost if it is discarded somewhere in the network between the two endpoints. Packets are lost in the network due to congestion or errors. Packets are late because the network delayed delivering the packet until its time for playout has passed.

Table 7 shows the percentages of packets discarded and lost for three cases: all tests, tests with acceptable call quality (CMOS ≥ 3.6) and tests with unacceptable call quality (CMOS < 3.6). We also show the percentage of packets that are lost or discarded, as well as the duplicate and out of order packet percentages. Out of order packets do not in general affect VoIP because the playout buffer reorders the voice samples into their proper order. Similarly, duplicate packets only waste bandwidth. They do not affect VoIP quality directly.

To achieve acceptable call quality, the sum percentage of lost and discarded packets must be kept less than 1%.

Table 7. Lost, Discarded, Duplicate, and Out of Order Packets

	All Tests	Acceptable CMOS ≥ 3.6	Unacceptable CMOS < 3.6
Late Packet Discards%	1.03%	0.72%	2.37%
Lost Packets%	0.43%	0.12%	1.78%
All Unplayed Packets%	1.46%	0.83%	4.15%
Out of Order Packets%	0.24%	0.17%	0.53%
Duplicate Packets%	0.06%	0.03%	0.17%

Voice Latency. Voice latency affects conversational quality in two ways. First, callers hear the echo of their own voice, and if the delay is too large, the echo becomes noticeable and disruptive. Second, long voice delay interferes with the ability of two people to know when to speak leading to collisions in the conversation.

Voice latency, *L*, comprises: codec delay, *C*, network delay, *N*, and playout delay which is also known as jitter buffer delay, *B,* in both directions as shown in Fig. 2 and the following formula.

$$L = C_u + N_u + B_u + C_d + N_d + B_d \tag{2}$$

The codec delays C_u and C_d are fixed by the choice of codec. For G.711, it is 20 milliseconds. For G.729 it is 30 milliseconds.

Network delays vary depending on the propagation delays due to distance, transmission delays due to link speed and packet size, forwarding delays in the routers, and queueing delays due to congestion. Network delays are measured using RTCP. RTCP packets are sent periodically during the call in both directions. Each RTCP packet *i* includes a timestamp T_i when it was sent. The receiver stores T_i and the time it received that packet R_i. When the receiver next sends RTCP packet *j*, it includes within the RTCP packet T_i, T_j-R_i, and T_j. When the sender receives that packet, it notes the time it received the packet R_j, and computes the network round trip delay using the formula:

$$(N_u + N_d) = (R_j - T_i) - (T_j - R_i) \tag{3}$$

The jitter buffer is a mechanism used to deal with variation in network latency. The delay introduced by jitter buffers B_u and B_d can be fixed if a fixed size jitter buffer is used or can vary if an adaptive jitter buffer is used. TestYourVoIP implements an adaptive jitter buffer based on [3]. The verifiers measure the average jitter buffer delay and include the measured value in the voice latency they report.

Long voice round trip delay negatively impacts conversational voice quality. Its affect on Average CMOS accounted for 8.5% of the degradation. The average voice round trip latency measured was 263 milliseconds. For calls with low voice quality (CMOS < 3.6), the effect of delay increases to 22% of the total degradation. Average voice round trip latency has increased to 467 ms for the 18% of the test calls with unacceptable voice quality.

Jitter. A packet is discarded when it arrives too late to be played out by the receiver. Packets arrive too late if they are delayed more than the playout buffer can accommodate. At the beginning of each talk spurt, the jitter buffer duration and size is adjusted based on observed values of delay and delay variation for earlier packets. The playout time for each incoming packet is computed based on its transmit time and the current jitter buffer duration. Packets that are delayed in network more than the jitter buffer can accommodate will arrive after their scheduled playout time and so will be discarded as too late. While some jitter buffer implementations have a finite size, for TestYourVoIP, we have set up the jitter buffer to be of unlimited size and so we never have any early packet discards.

Average jitter is a measure of the variation in network delay and is relatively easy to compute. Average jitter is often quoted in Service Level Agreements as a limit on the variation in delay experienced by packets. Average upstream jitter J_U, downstream jitter J_D, and total jitter J for a call are defined as:

$$J_U = \frac{\sum_{i=1}^{n-1} \left|(V_{i+1} - U_{i+1}) - (V_i - U_i)\right|}{n-1}, \quad J_D = \frac{\sum_{i=1}^{n-1} \left|(Y_{i+1} - W_{i+1}) - (Y_i - W_i)\right|}{n-1}, \quad J = (J_U + J_D)/2 \tag{4}$$

The jitter measured for all test calls, and calls with acceptable and unacceptable call quality are shown in Table 8. The measured average jitter is relatively small, 6.8 milliseconds for all tests. When we compare tests that achieved an acceptable voice quality with tests with unacceptable voice quality, we find the average jitter only increases from 6.1 to 9.9 milliseconds. Note that the average delay increases by roughly the same percentage. The ratio of jitter over delay stays roughly the same.

The data hints that the average jitter is not a very good measure for predicting whether or not a network can support VoIP. Instead we should examine late packet discards since they directly impact voice quality. Late packet discards takes into account the distribution of network delays, not just the average.

Table 8. Jitter Measurements

	All Tests	**Acceptable CMOS ≥ 3.6**	**Unacceptable CMOS < 3.6**
Upstream Jitter J_U	7.3 msec	6.4 msec	11.2 msec
Downstream Jitter J_D	6.2 msec	5.7 msec	8.6 msec
Total Jitter J	6.8 msec	6.1 msec	9.9 msec
Voice Delay	263 msec	216 msec	467 msec
Jitter/Delay Ratio	0.026	0.028	0.021
Late Packet Discards	1.03%	0.72%	2.37%

4.4 Codec Comparison

TestYourVoIP users can select the codec used in their test calls. While most choose the default G.711 codec, slightly more than 35,000 tests were run using the low bandwidth G.729 codec. Table 9 compares the results obtained for the two codecs. Because the G.729 codec degrades the voice quality more than the G.711 codec, it is not surprising that the average CMOS of the G.729 codec is less than that of G.711. However, one would hope that the G.729 codec would perform better under more challenging network conditions due to its decreased bandwidth. Fig. 4 shows the distribution of CMOS values for both codecs. The G.711 codec is more likely to outperform the G.729 codec through most of the range. Only when the CMOS score drops to below 1.4 is the G.729 codec more likely to have a better CMOS than that score. This only accounts for 0.3% of all tests.

Table 9. Codec Comparison

G.711	**All Tests**	**Acceptable CMOS ≥ 3.6**	**Unacceptable CMOS < 3.6**
Average CMOS	3.97	4.19	2.93
Voice Delay	256 msec	214 msec	457 msec
Lost Packets	0.43%	0.12%	1.92%
Late Packet Discards	1.01%	0.72%	2.38%
$ACQ_{3.6}$	82.6%		
Tests with bad network	8.2%		
G.729			
Average CMOS	3.63	3.97	2.94
Voice Delay	327 msec	235 msec	510 msec
Lost Packets	0.43%	0.09%	1.10%
Late Packet Discards	1.19%	0.63%	2.31%
$ACQ_{3.6}$	66.7%		
Tests with bad network	16.8%		

These results suggest that the conditions where the G.729 codec outperforms the G.711 codec are rare. Given all the work done to develop low bandwidth codecs, we are surprised at this result.

4.5 Trends

Call quality over 2005 has generally been trending downwards as shown in Fig 5. When we fit linear least squares trends to the data, we find CMOS is decreasing by 0.007 points per month and ACQ is decreasing by 0.4% per month with R^2 values of 0.66 and 0.70 respectively. We find the trend in Acceptable Call Quality to be particularly distressing. If this trend continues, each year 5% more of the VoIP calls on the Internet will experience unacceptable call quality.

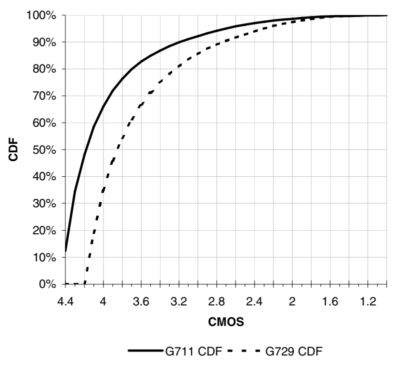


Fig. 4. Comparing CMOS for G.711 and G.729

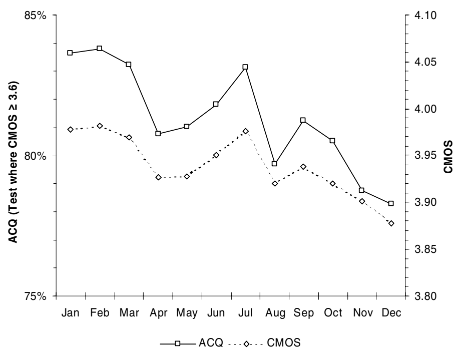


Fig. 5. Call Quality Trends in 2005

5 Future Work

TestYourVoIP has collected a rich dataset describing VoIP quality on the Internet. In this paper we have only scratched the surface of the lessons this data can teach us. Some other questions we can explore are:

- Does voice quality vary by time of day or day of the week?
- Is voice quality symmetric?
- Is voice quality the same worldwide, or are there geographic differences?
- How does voice quality vary by distance, hops, and route?
- How do different jitter buffer strategies perform?
- What the impact of the components of voice round trip latency?
- What is the cause of the differences seen in voice quality between codecs?
- How does average jitter statistically relate to late packets and voice quality?

6 Conclusions

While average CMOS on the Internet in 2005 was 3.94, only 81% of the test calls achieved an acceptable call quality of 3.6 ($ACQ_{3.6}$). Far too many calls had unacceptable call quality.

Of the three causes of degraded voice quality that network providers can affect; late packet discards, lost packets, and voice latency, the most important is late packet discards. However, if we focus on only those calls with unacceptable call quality, the pattern changes, and the three causes have roughly equal weights. Providers cannot focus on improving just one cause of poor voice quality to improve $ACQ_{3.6}$, rather, they must work to improve all three causes.

The data hints at two surprising results.

1. Average Jitter, a metric often used in service level agreements does not seem to be a good predictor of voice quality. A better choice for service level metrics is Late Packet Discards, a metric that directly affects voice quality.

2. When we compare the voice quality that codecs actually achieve on the Internet, we find that the G.711 codec is more likely to produce good voice quality than the low bandwidth G.729 codec. While 82.6% of the tests with the G.711 codec had acceptable call quality, only 66.7% of the tests with the G.729 codec had acceptable call quality.

Finally, the data clearly shows a slow but clear decrease in voice quality during 2005. Voice Quality on the Internet is getting worse.

References

1. The E Model: A computational model for use in transmission quality, ITU-T Recommendation G.107, Dec 1998.
2. Telchemy, Inc. Voice Quality Measurement, Jan, 2005, http://www.telchemy.com/appnotes/TelchemyVoiceQualityMeasurement.pdf.
3. Ramjee. R., Kurose, J., Towsley, D., and Schulzrinne, H., "Adaptive playout mechanisms for packetized audio applications in wide-area networks", in Proceedings of the Conference on Computer Communications (IEEE Infocom), Toronto, Canada, June 1994, pp. 680-688.
4. Markopoulou, A., Tobagi, F., and Karam, M., "Assessing the Quality of Voice Communications Over Internet Backbones", IEE/ACM Trans. Networking, v. 11, no. 5, Oct 2003.
5. Claffy, K.C., Crovella, M., Friedman, T., Shannon, C. and Sring, N., "Community-Oriented Network Measurement Infrastructure (CONMI) Workshop Report", ACM SSIGCOMM Comp. Comm. Review, V. 36, No. 2, Apr 2006.
6. CAIDA, the Cooperative Association for Internet Data Analysis, http://www.caida.org/
7. Simpson, Jr., C. and Riley G., "NETI@home: A Distributed Approach to Collecting End-to-End Network Performance Measurements", PAM2004 - A workshop on Passive and Active Measurements, Apr 2004.
8. Shavitt, Y. and Shir, E., "DIMES: A Distributed Architecture for Internet Measurement and Monitoring", IEEE Infocom 2005.
9. Jiang, W. and Schulzrinne, H. "Assessment of VoIP Service Availability in the Current Internet", PAM 2003 A workshop on Passive and Active Measurements.
10. "Keynote's Hard Numbers On VoIP Quality", VOIP Magazine, July 2005, http://www.voip-magazine.com/content/view/275/

A WSDM-Based Architecture for Global Usage Characterization of Grid Computing Infrastructures

Glauco Antonio Ludwig[1], Luciano Paschoal Gaspary[2],
Gerson Geraldo Homrich Cavalheiro[1], and Walfredo Cirne[1]

[1] Universidade do Vale do Rio dos Sinos (UNISINOS), Brazil
{glaucol, gersonc}@unisinos.br
[2] Universidade Federal do Rio Grande do Sul (UFRGS), Brazil
paschoal@inf.ufrgs.br
[3] Universidade Federal de Campina Grande (UFCG), Brazil
walfredo@dsc.ufcg.edu.br

Abstract. Current solutions to characterize grid computing usage are limited in three important aspects. First, they do not provide a global, uniform view of the use of infrastructures comprised of heterogeneous grid middleware. Second, they do not allow the specification of policies to publicize the collected information. Third, they do not generate statistics about the applications that are executed on the grid. To fill this gap, we propose an architecture based on the Web Services Distributed Management standard and on access control policies to characterize global usage of grid computing infrastructures, even when such grids are formed by heterogeneous middleware packages. We introduce this architecture and present preliminary results obtained with a prototype.

1 Introduction

The lack of solutions to manage complex systems such as grid computing infrastructures has hampered the widespread use of this technology, specially in the corporate arena. When used in large scale, involving many institutions and participants, it is necessary – in addition to managing faults, configuration, performance, and security – to characterize the use of the grid infrastructure. The objective is sixfold: (*i*) obtain detailed information about the applications executed on the grid (i.e. where they were executed, execution duration, resources consumed, and users who submitted them); (*ii*) identify the execution of malicious applications (an application executing for a very long time could indicate the intention of only consuming grid resources, impeding their legitimate use); (*iii*) determine users who contribute resources for the grid and those who consume most of the computing power available; (*iv*) guarantee a fair scheduling of jobs on the grid, denying or allowing users to execute new applications based on their usage history (users that exceed a usage quota should not have access to the resources of the grid); (*v*) identify stations of the grid which are not contributing in a productive way (a certain station receives jobs to compute and ends up failing to process them very often); and (*vi*) follow the evolution of the grid computing infrastructure (allowing the recognition of usage patterns and trends).

Currently, there are several solutions [2, 4, 9, 11, 12, 14, 17] that provide the administrator with statistics about the use of grid systems. Most of them are limited to monitor the status of environment resources, neglecting statistical and historical data about the execution of applications on the grid. For example, with the existing solutions it is possible to observe that a certain station had its CPU highly loaded for the past twelve hours. However, one cannot precisely infer the reason for that. In short, these solutions do not relate in a proper way information about the resources and the applications running on top of them.

It is usual to have heterogeneous grid middleware being employed in a large scale, inter-institutional grid computing infrastructure. Each of them (e.g. Globus [7], Condor [3], and OurGrid [13]) uses its own indicators to report job execution. In this context, a problem to be overcome by a characterization and accounting solution is the use of a common format to represent basic accounting and usage data. Existing tools do not handle well such heterogeneity and, therefore, are unable to provide the grid administrator with a uniform, integrated view of the usage of a heterogeneous grid setup.

While institutions involved in a grid computing infrastructure are willing to share usage information with its collaborating sites, each institution has a different security policy to be applied. Some of the data gathered by a characterization and accounting tool can be classified (e.g. details of the station that executed a job) and, therefore, their distribution conflicts with the security policy in place. Here, again, there is no support in current solutions to define and enforce a policy for the distribution of data to the sites comprising the grid.

Considering the above limitations, we propose an architecture to support global usage characterization of grid computing infrastructures composed of different middleware packages such as Globus, Condor, and OurGrid. Our architecture allows the grid administrator to obtain long-term statistics and real-time notifications about major events generated by both the grid resources and applications. Currently, we are focusing on managing the grid applications, filling a not yet explored gap. In addition, we propose a mechanism to allow each institution participating of the grid infrastructure to specify and enforce policies for the publication of usage information generated inside the institution.

In line with current grid technologies, whose components have been organized as web services, the architecture relies on the recent OASIS Web Services Distributed Management (WSDM) standard [15]. Despite the fact this work focuses on the issue of accounting, we regard it as a building block to achieve a more scalable and autonomous management solution, based on the composition/choreography of complementary WSDM-compliant grid management services.

The remainder of the paper is organized as follows. Section 2 discusses related work on usage characterization of grid computing infrastructures. Section 3 introduces the architecture and Section 4 details its components. Section 5 presents preliminary results obtained with a prototype. Section 6 closes the paper with concluding remarks and perspectives for future work.

2 Related Work

Important steps have been taken towards characterization of grid computing infrastructures, with relevant mechanisms being proposed in the past to address specific issues such as distributed resource usage monitoring. Nevertheless, they are not fully prepared to (*i*) provide a global, uniform view of the use of infrastructures comprised of heterogeneous grid middleware, (*ii*) allow the specification of policies to publicize the collected information, and (*iii*) generate statistics about the applications executed on the grid.

Solutions such as Remos [4], visPerf [9], GridRM [2], MonALISA [12], and Ganglia [11] are unable to operate over heterogeneous infrastructures, which share re-sources employing distinct grid computing middleware packages. Since each middleware tends to adopt a proprietary format to represent statistics about applications executed and resources consumed, they are not prepared to collect such statistics and normalize them using a uniform information model. Due to this limitation, it becomes a challenge to provide the grid administrators with a global view of the grid usage.

Regarding selective dissemination of grid usage information, just a few systems – such as visPerf and GridRM – provide access control mechanisms. These mechanisms, however, are limited. visPerf, for example, employs only one credential for the whole grid; users that possess this credential can access all information generated by the infrastructure. GridRM adopts an access control mechanism based on administrative domains. When an institution decides to make part of a grid infrastructure that uses GridRM, it starts to publicize – to the domains of interest – all the information generated by the monitoring agents located within its administrative boundaries. In this context, we claim that neither of the approaches are flexible enough to handle the definition and enforcement of information publicization policies.

MonALISA and Usage Record (UR) [10] are the only solutions that provide statistics related to the execution of applications on the grid. The former does not relate such statistics with the resources consumed, hampering a precise understanding of the grid usage. The latter constitutes an information model that merges data from both applications executed and resources consumed. Since it is not a software architecture, UR does not define how this data must be collected, processed, consolidated, and presented to the grid administrator.

3 Architecture Overview

The architecture proposed is composed of four major components: *Publishers*, *Characterization Service*, *Authorization Service*, and *Management Application*. Figure 1 illustrates a general view of the architecture and the relations between its components.

Publishers are grid-technology-dependent piece of software responsible for monitoring the occurrence of predefined events generated by the components comprising the grid (e.g. a log message informing that a certain application has been submitted). Whenever such an event is observed, the publisher extracts the data of interest from the event, normalizes and send it to the characterization service. Publishers are supposed to be developed and distributed across the infrastructure, in accordance to the needs of every grid middleware taking part of the overall inter-institutional setup. For example, supposing a scenario composed of two distinct grid middleware packages, say Globus (domain A) and OurGrid (domain B), specific Globus and OurGrid publishers should be installed in both domains – exactly in the location where data about the resources and the running applications is generated.

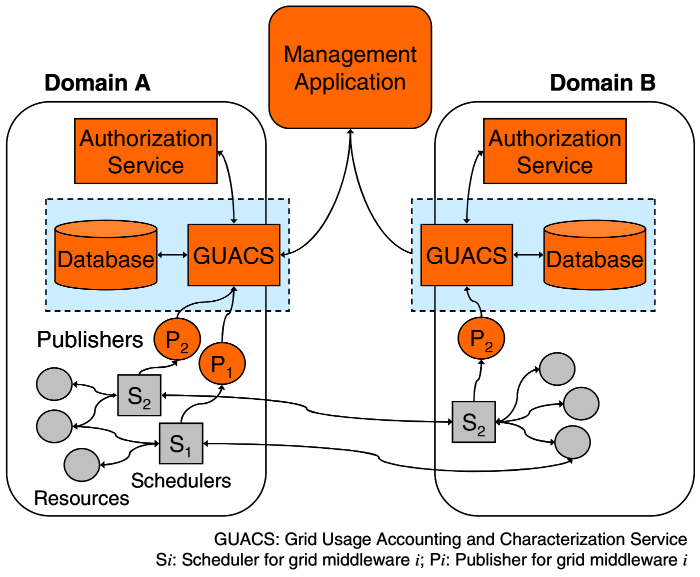


Fig. 1. Conceptual view of the architecture

The *Characterization Service* is a software entity, whose purpose is, among other, to receive data from the publishers and store them in a local, normalized database. By normalized database we mean that regardless of the grid middleware being monitored, the database always stores the same data (i.e. the information model is the same). To address potential scalability problems, the service can be instantiated in several locations (e.g. per institution, per department, etc.); important is to cover the whole grid infrastructure.

In addition to receiving and processing data sent by the publishers, the characterization service also serves a management application (or other management services), which may request for historical and/or real-time information about the resources and the applications running on the grid. Two types of requests are currently supported: *publish/subscribe* and *query/response*.

Whenever a request is received by the characterization service, it is first passed to an *Authorization Service*, which is responsible for granting or not the former permission to reply the request. Based on a role-based access control policy [5], it determines whether or not (and to what extent) a request originated from a certain institution can be replied.

The architecture is also composed of a *Management Application*, from where the administrator can subscribe for/be notified of relevant events generated by the whole grid infrastructure. He/she can also gather historical information from the characterization service instances and plot several graphs in order to characterize grid usage. Note that, due to the normalized information model employed by the characterization service, all the data gathered – regardless of the grid system used in every institution – has the same format. Therefore, it is possible to draw a picture of the whole grid infrastructure usage in an integrated, uniform way. This *complete view* of the grid will may be compromised if the policy to distribute information employed by the institutions making part of the grid is too restrictive.

It is possible to have several instances of the management application running simultaneously (e.g. one per domain or department). Each of them will present a more or less detailed view of the grid infrastructure, depending on the permissions that the requester has in the characterization services.

4 Components of the Architecture and Implementation

In this section we describe the components of the architecture in more detail, focusing on design and implementation decisions. Recall that the architecture stands on the WSDM specification. Therefore, other associated standards such as WS-Notification [8] and WSRF (Web Services Resource Framework) [6] are also employed.

Table 1. Usage Record format

Field name	Description
Username	User's login name corresponding to user Id in /etc/passwd file.
ProjectNam e	Name/identifier of the project or charge group associated with this usage.
JobId	Identifier of the job.
Queue	Name of the queue from which the job was executed or submitted.
GridId	User's global unique Id. Distinguish Name in the user's X509 certificate.
FromHost	Name of the host from which the job was submitted.
Host	Name of the host on which the job ran.
StartTime	Date when the job started running in date time format (UTC time zone).
EndTime	Date when the job completed in date time format (UTC time zone).
Processors	Number of processors either used or requested that each center uses (for billing purpose).
NumNodes	Number of nodes used.
CpuTime	CPU time used, summed over all processes in the job.
WallTime	Wall clock time elapsed while the job was in the running state.
Memory	Maximum amount of virtual memory used by all concurrent processes in the job.
Disk	Disk storage used.
Network	Network used (withdrawals) or requested (reservations).
JobName	Job or Application name.
Status	Number representing completion status of the job.
Charge	Total charge of the job in system's allocation unit.

4.1 Information Model

As already mentioned in the previous section, we have adopted a uniform information model to be used by all instances of the characterization service. We use the UR (Usage Record) model [10], proposed by the Usage Record Working Group of the Global Grid Forum. This model is the result of an effort to define a common usage record based on those employed in current grid sites.

Table 1 illustrates the fields included in the Usage Record. As one can see, it merges information related to resource usage (e.g. Processors and NumNodes) and applications (e.g. JobName and Status).

4.2 Publishers

For every grid middleware there must exist a specific publisher, which is able to collect data about the resources and the running applications on that particular system. In our current implementation of Globus and OurGrid publishers, this data is gathered from the log files generated by both grid systems.

Whenever a new event is detected in the grid system (e.g. job started and job completed), the publisher executes a SETRESOURCEPROPERTIES request to the *Characterization Service* (to where it is virtually attached) updating a *resource property* defined within the service (e.g. JOBSTARTED and JOBCOMPLETED). The content of such a request is illustrated in figure 2a and b. Note that each *set* operation comprises the update of several data into the service (through the use of complex types). A simplified XML document is used to express the updates required by each possible SETRESOURCEPROPERTIES request. In addition, as it is the case of the examples presented, the data updated by a set request to a certain property (e.g. JOBCOMPLETED) is complementary to the data supplied by the set request to other property (e.g. JOBSTARTED).

```
<JobStarted>
  <Username> glauco_ludwig </Username>
  < ProjectName> bio_paua </ProjectName>
  <JobId> 1.1.1.node01.unisinos.br </JobId>
  <Queue> node01 </Queue>
  <GridId> glauco_ludwig@unisinos.br </GridId>
  <FromHost> node01.unisinos.br </FromHost>
  <Host> node09.unisinos.br </Host>
  <StartTime> 2005-09-13 17:24:50 </StartTime>
  <JobName> intracel_dynamics </JobName>
</JobStarted>
```

(a) JobStarted

```
<JobCompleted>
  <JobId> 1.1.1.node01.unisinos.br </JobId>
  <EndTime> 2005-09-13 17:30:22 </EndTime>
  <Processors> 2 </Processors>
  <NumNodes> 1 </NumNodes>
  <CpuTime> PT15S </CpuTime>
  <WallTime> PT45M48S </WallTime>
  <Memory> 1234 </Memory>
  <Disk> 560 </Disk>
  <Network> 1.000.000 </Network>
  <Charge> 300 </Charge>
</JobCompleted>
```

(b) JobCompleted

Fig. 2. Format of a SETRESOURCEPROPERTIES request

4.3 Characterization Service

The characterization service operates as an intermediary component between the publishers and the management application. The communication of both publishers and the management application with the service is carried out through a WSDM interface.

Publishers update data into the service through SETRESOURCEPROPERTIES requests, while a management application can (*i*) request for the value of a single property (GETRESOURCEPROPERTY), (*ii*) request for a filtered set of information – through an XPath query – of one or more resource properties (QUERY RESOURCEPROPERTIES), and (*iii*) subscribe for a resource property, as well as receive notifications. Figures 3 and 4 illustrate part of the SOAP envelope of both a subscription to and a notification of change in the resource property JOBFAILED.

```
<wsnt:Subscribe>
  <wsnt:ConsumerReference>
    <wsa:Address> http://www.unisinos.br:8080 </wsa:Address>
    <wsa:ReferenceProperties/>
  </wsnt:ConsumerReference>
  <wsnt:TopicExpressionDialect=”http://docs.oasis-open.org/wsn/2004/06/TopicExpression/Simple”>
    cs:JobFailed
  </wsnt:TopicExpression>
</wsnt:Subscribe>
```

Fig. 3. Format of a WSN-based subscription to JOBFAILED resource property

```
<wsn:Message>
  <wsrf:ResourcePropertyValueChangeNotification xmlns:wsrf=”http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-
   ResourceProperties-1.2-draft-01.xsd”>
    <wsrf:OldValue>
      <cs:JobFailed xmlns:wsrp=”http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.
       Xsd” xmlns=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:fs=”http://ws.apache.org/resource/gaems”>
        …
      </cs:JobFailed>
    </wsrf:OldValue>
    <wsrf:NewValue>
      <cs:JobFailed xmlns:wsrp=”http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.
       Xsd” xmlns=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:fs=”http://ws.apache.org/resource/gaems”>
        <fil:JobId> 2.1.1.node02.unisinos.br </fil:JobId>
        <fil:EndTime> 2005-09-13 15:30:24 </fil:EndTime>
        <fil:Processors> 4 </fil:Processors>
        <fil:NumNodes> 4 </fil:NumNodes>
        <fil:CPUTime> PT15S </fil:CPUTime>
        <fil:WallTime> PT45M48S </fil:WallTime>
        <fil:Memory> 5000 </fil:Memory>
        <fil:Disk> 850 </fil:Disk>
        <fil:Network> 10.000.000 </fil:Network>
        <fil:Charge> 0 </fil:Charge>
      </cs:JobFailed>
    </wsrf:NewValue>
  </wsrf:ResourcePropertyValueChangeNotification>
</wsn:Message>
```

Fig. 4. Format of a WSN-based notification for the JOBFAILED resource property

With reference to the resource properties available in the characterization service, currently six properties are offered: JOBSTARTED, JOBCOMPLETED, JOBABORTED, JOBFAILED, JOBCANCELED, and JOBHISTORY. These properties are detailed in table 2.

Note that the first fifth properties are expected to be subscribed for. Hence, when a job is completed, successfully or not, the management application may receive a real-time notification, which can be used either by the grid system itself (e.g. to reschedule a failed job) or the administrator. The JOBHISTORY property, on the other hand, stores multiple instances of the Usage Record, which can be retrieved by a management application through GETRESOURCEPROPERTY or QUERYRESOURCEPROPERTIES requests.

The characterization service has been implemented based on the Muse framework [1] from the Apache Software Foundation.

Table 2. Resource properties available in the characterization service

Resource property	Operations supported	Information available
JobStarted	Subscribe/publish	Username, ProjectName, JobId, Queue, GridId, FromHost, Host, StartTime, JobName
JobCompleted, JobAborted, JobFailed, JobCanceled	Subscribe/publish	JobId, EndTime, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, Charge
JobHistory	GetResourceProperty, QueryResourceProperties	Username, ProjectName, JobId, Queue, GridId, FromHost, Host, StartTime, EndTime, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, JobName, Status, Charge[1]

[1] The information available matches those of the Usage Record. The JobHistory property stores multiple instances of the record.

4.4 Authorization Service

The authorization service is invoked by the characterization service whenever it receives a request from a management application. Depending on the identification of the requesting institution and the policies defined by the owner of the characterization service, three situations may occur: a response is not sent; a response with partial content is sent; or a complete response is sent.

The authorization model employed by the service to specify and enforce information distribution policies is the RBAC (Role-Based Access Control) [5]. RBAC was chosen because it is widely accepted and simplifies the management of policies. In this format, policies (e.g. permissions and restrictions) are associated to roles, and institutions are assigned to the proper roles.

Table 3 illustrates the organization proposed for the policy repository. Institutions may belong to one or more roles. Each role, on its turn, comprises the set of the Usage Record fields allowed to be distributed.

Table 3. Example of policies specified for a hypothetical institutional characterization service

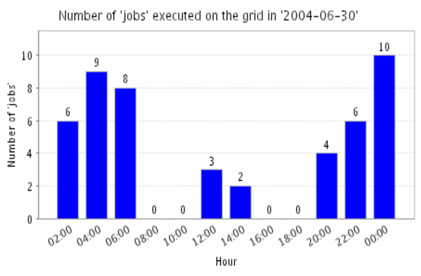
Host/institution	Role	Policies[1]
University A	University	JobStarted, JobCompleted, JobAborted, JobFailed, JobCanceled, JobHistory Username, ProjectName, JobId, Queue, GridId, FromHost, Host, StartTime, EndTime, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, JobName, Status, Charge
Company B	Enterprise	JobHistory JobId, GridId, FromHost, StartTime, EndTime, CpuTime, JobName, Status, Charge
Company C	Enterprise	JobHistory JobId, GridId, FromHost, StartTime, EndTime, CpuTime, JobName, Status, Charge
Other	Default	None

[1] Information allowed to be distributed.

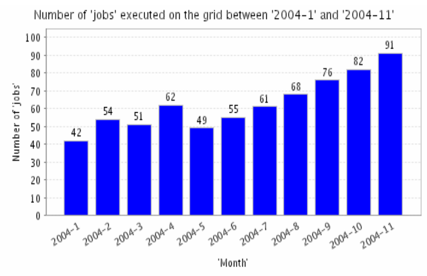
4.5 Management Application

In the management application, the grid administrator can configure the characterization services it is going to interact with, as well as dynamically subscribe for/be notified of relevant events generated by them. In the case of the notifications received, they are presented in an event console. As for the historical information retrieved from one or more characterization service instances, several plots can be automatically generated, providing an integrated view of the usage of the grid computing infrastructure (see figure 5 for a simple example).

Our current prototype was implemented in Linux, using the Java programming language, the HSQLDB database, and the JFreeChart library (for the generation of plots). We are also working on the design and implementation of a release of the management application to run integrated to the HP Open View management platform.



(a) Short-term



(b) Long-term

Fig. 5. Plot of job execution statistics

5 Experimental Evaluation

To prove concept and technical feasibility of the architecture, we have instantiated it in a real setup, composed by three administrative domains: A, B, and C. In two of them, A and B, both Globus and OurGrid middleware packages were deployed, while in domain C only OurGrid was used. One instance of the *Characterization Service* was installed and executed in each domain. In this environment, we were able to evaluate the impact of policy definition on the dissemination of grid usage data among the domains (explored in a paper being currently prepared by our research group). In addition, we were able to stress both the *publish/subscribe* mechanism (properties JobStarted, JobCompleted, JobAborted, JobFailed, and JobCanceled) and the requests for *historical* data (property JobHistory). Data retrieved by requesting the latter allowed the management application to draw plots offering an integrated view of the whole grid infrastructure.

A preliminary performance evaluation of the architecture has also been carried out, restricting the experiment to one *Publisher*, one instance of the *Characterization Service,* and the *Management Application*. Each of these components was executed in a different PC with 2 Pentium4 2.4 GHz processors, 1 GB of RAM memory, and GNU/Linux Red Hat 8.0 operating system, which were interconnected through a 100 Mbps switch.

To obtain statistically sound results, each experiment was repeated 400 times. The *end-to-end delay*, i.e. the time interval measured between the moment when an event is generated by the grid middleware and the moment when it is reported to the management application, was in average 245.62 ms. The standard deviation of the observed measurements was 3 ms. Although this a hardware-dependent result, it represents a good estimative of what one can expect in terms of the processing overhead imposed by the management plane. We consider this result acceptable, since the management application will be almost immediately informed about the occurrence of important events and will be able to react, in a timely manner, to interruptions in the execution of applications.

6 Conclusion and Future Work

The use of grid computing infrastructures is consolidated in academic environments, but in corporate environments their deployment is not as accelerated as originally expected. We attribute this to the lack of security and, specially, management mechanisms to provide a reliable, secure, and controllable grid computing environment. In this paper we proposed an architecture – based on a management standard highly conformant with the current web services orientation of grid technologies – to characterize and account, in a uniform and integrated way, usage of grid computing infrastructures composed of heterogeneous middleware packages.

The architecture allows the grid administrator, in addition to receiving real-time notifications about major events, to obtain long-term statistics about jobs executed, resources used, and so on. This information allows one to precisely characterize grid usage (e.g. in terms of top grid consumers, types of jobs executed, stations most used). Existing solutions are limited, since they only gather complementary information such as CPU load and memory use of the grid machines. From a grid management perspective, the statistics the architecture is able to provide are important (*i*) to assess the volume of accesses to the grid infrastructure, the communications established, (*ii*) to draw a global grid usage profile, and (*iii*) to optimize and plan the capacity of the grid.

Another important aspect of the architecture to be highlighted is its ability to selectively distribute information taking into account the policies defined by every institution comprising the grid computing infrastructure. Although a complete view of the grid may not be offered if the policies are too restrictive, we believe it can be avoided through negotiation and agreements between the institutions.

A major contribution of this work relies on the usage of WSDM, which consists of a common approach for managing all components of a grid environment, including resources and services. As far as we are aware, this is one of the first research papers to propose (and report a real implementation of) a grid management service compliant with this recently standardized specification.

As future work we intend to better evaluate the architecture developed, stressing it under different number of (and simultaneous) publishers, subscriptions, and policies. Besides, we will extend the architecture with complementary services to support additional grid management functionality (fault, configuration, performance, and security), following a *plug-and-play* design.

Acknowledgments

This work has been developed in collaboration with HP Brazil R&D.

References

1. Apache Web Services – Muse Project Home Page (2006). http://ws.apache.org/muse/.
2. Baker, M. and Smith, G. (2003). GridRM: An Extensible Resource Monitoring System. IEEE International Conference on Cluster Computing, pp. 207-215.
3. Condor Project Home Page (2005). http://www.cs.wisc.edu/condor.
4. Dinda, P., Gross, T., Karrer, J. et al. (2001). The Architecture of the Remos System. IEEE International Symposium on High Performance Distributed Computing, pp. 383-394.
5. Ferraiolo, D. F., Sandhu, R., Gavrila, S. et al. (2001). Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security, v. 4, n. 3, pp. 224-274.
6. Foster, I., Czajkowski, K., Ferguson, D. E. et al. (2005). Modeling and Managing State in Distributed Systems: the Role of OGSI and WSRF. Proceedings of the IEEE, v. 93, issue 3, pp. 604-612.
7. Globus Toolkit Home Page (2005). http://www.globus.org.
8. Graham, S., Hull, D., and Murray, B. (2006). Web Services Base Notification 1.3 (WS-BaseNotification). OASIS Public Review Draft. http://www.oasis-open.org/committees/download.php/18546/wsn-ws_base_notification-1.3-spec-pr-03.doc.
9. Lee, D., Dongarra, J., and Ramakrishna, R. et al. (2003). VisPerf: Monitoring Tool for Grid Computing. International Conference on Computational Science, pp. 233-243.

10. Mach, R., Lepro-Metz, R., and Jackson, S. (2005). Usage Record – Format Recommendation. Global Grid Forum Usage Record Working Group. http://www.psc.edu/ ~lfm/ PSC/Grid/UR-WG/UR-Spec.v1.pdf.
11. Massie, M. L., Chun, B. N., and Culler, D. E. (2004). The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. Parallel Computing, v. 30, n. 7, pp. 817-840.
12. Newman, H. B., Legrand, I. C., Galvez, P. et al. (2003). MonALISA: A Distributed Monitoring Service Architecture. Computing in High Energy and Nuclear Physics.
13. OurGrid Project Home Page (2005). http://www.ourgrid.org.
14. Tierney, B., Aydt, R., Gunter, D. et al. (2002). A Grid Monitoring Architecture. Global Grid Forum Performance Working Group. http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-16-3.pdf.
15. Vambenepe W. (2005). Web Services Distributed Management: Management Using Web Services (MUWS 1.0) Part 1. OASIS Standard. http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part1-1.0.pdf.
16. Vinoski, S. (2005). Web Service References. IEEE Internet Computing, v. 9, no. 3, pp. 90-93.
17. Wolski, R., Spring, N., and Hayes, J. (1999). The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. Journal of Future Generation Computing Systems, v. 15, n. 5-6, pp. 757-768.

Management of DiffServ-over-MPLS Transit Networks with BFD/OAM in ForCES Architecture*

Seung-Hun Yoon, Djakhongir Siradjev, and Young-Tak Kim**

Dept. of Information and Communication Engineering,
Graduate School, Yeungnam University
214-1, Dae-Dong, Kyungsan-Si, Kyungbook, 712-749, Korea
bthuni@yumail.ac.kr, m0446086@chunma.yu.ac.kr, ytkim@yu.ac.kr

Abstract. This paper proposes a management of DiffServ-over-MPLS transit network with BFD(Bidirectional Forwarding Detection)/OAM (operation, administration and maintenance) in ForCES (Forwarding and Control Element Separation) architecture for QoS-guaranteed DiffServ-over-MPLS traffic engineering. The proposed BFD and ForCES functions are implemented on Intel 2400 network processor, where BFD/OAM packets for MPLS TE-LSP are exchanged every 5 ~ 10 ms interval for performance measurements and link failure detection. The operations of BFD/OAM-based fault detection and performance measurement are controlled via distributed control plane with ForCES (forwarding and control element separation) architecture for large scale IP/MPLS router using multiple network processors in each network interface card. We explain the implementation details of ForCES-based distributed control plane functions, hierarchical traffic grooming with label stacking, and BFD/OAM mechanisms. The link failure detection performance of BFD/OAM functions for MPLS TE-LSP is evaluated.

Keywords: DiffServ-over-MPLS, QoS, ForCES, BFD, OAM, Network Processor.

1 Introduction

In next generation Internet, various QoS-guaranteed realtime broadband multimedia services, such as video telephony, multimedia teleconference, IP-TV and video-on-demand, should be provided based on IP/DiffServ-over-MPLS transit networks with efficient traffic engineering [1]. For end-to-end QoS-guaranteed multimedia service provisioning, the virtual overlay transit network for each DiffServ class-type must be continuously monitored for available bandwidth and edge-to-edge packet delivery performance, such as delay, jitter, and packet loss/error rate[1].

IETF BFD (Bidirectional Forwarding Detection) has been designed to detect faults in the bidirectional path between two forwarding entities with protocol independent to physical layer and path types [2-5]. BFD also provides the continuity checking

* This research was supported by the MIC, under the ITRC support program supervised by the IITA.
** Corresponding author.

functions of data link layer as OAM (operation, administration and maintenance), and the link management protocol (LMP) of WDM optical link. The most important function of BFD is protocol-independent fast detection of data link failure on any kind of path between two nodes, including direct physical links, virtual circuits, tunnels, MPLS LSPs, multi-hop routed paths, and uni-directional links, so long as there are some return paths. Except SONET/SDH transmission systems, fast link failure detection and fault restoration are not mostly supported by physical layer. In order to provide link failure detection and fault restoration within 50 ms (as in the automatic protection switching of SONET/SDH), the BFD/OAM continuity check must be performing at 5 ~ 10 ms interval, and dedicated hardware functions of network processor are required.

IETF ForCES (forwarding and control element separation) standards [6-9] aim to define a framework and associated mechanisms for exchange of information between the logically separate functionality of the control plane (including routing protocols, admission control, and signaling) and the forwarding plane (including fast packet processing such as packet forwarding, queuing, and header editing). The standard separation mechanism of ForCES allows the control plane and forwarding plane to innovate in parallel while maintaining interoperability [5]. In distributed/parallel IP/MPLS packet switching architecture, the control plane functions and the data forwarding plane functions should be carefully distributed to increase the processing capacity by parallelism while minimizing the inter-module communication overhead.

In this paper, we design and implement the management functions of DiffServ-over-MPLS transit network with BFD/OAM in ForCES architecture for QoS-guaranteed broadband realtime multimedia service provisioning. The proposed BFD and ForCES functions are implemented with Intel 2400 network processor, where BFD/OAM packets for MPLS TE-LSP are exchanged every 5 ~ 10 ms for performance measurements and link failure detection. The operations of BFD/OAM-based fault detection and performance measurement are controlled via distributed control plane with ForCES architecture for large scale IP/MPLS router using multiple network processors in each network interface card (NIC).

The rest of this paper is organized as follows. Section 2 describes the related work on BFD/OAM, ForCES, and distributed OSPF function. In Section 3, we explain the implementation of BFD/OAM functions for DiffServ-over-MPLS transit networks with hierarchical TE-Links. Section 4 analyzes the overall performance of the proposed BFD/OAM functions for DiffServ-over-MPLS transit networks, and finally we conclude this paper in section 5.

2 Background

2.1 Bidirectional Forwarding Detection

BFD is a protocol intended to detect faults in the bidirectional path between two forwarding engines, including physical interfaces, subinterfaces, data link(s), and to the extent possible forwarding engines themselves, with potentially very low latency[2-5]. It operates independent of transmission media, data protocols, and routing protocols. An additional goal is to provide a single mechanism that can be used for continuity checking and QoS measurement (including delay, jitter, and packet error/loss) over any transmission media, at any protocol layer, with a wide range of detection times and overhead, to avoid a proliferation of different methods.

BFD includes following important characteristics [2]: i) It must be simple, fixed-field encoding to facilitate implementations in hardware, ii) It should be independent of the data protocol being forwarded between two systems; BFD packets are carried as the payload of whatever encapsulating protocol is appropriate for the medium and network, iii) BFD must be path-independent. BFD can provide failure detection on any kind of path between systems, including direct physical links, virtual circuits, tunnels, MPLS LSPs, multi-hop routed paths, and unidirectional links, so long as there is some return paths. BFD also provides the continuity checking functions of data link layer as OAM, and the link management protocol (LMP) of WDM optical link.

2.2 ForCES (Forwarding and Control Element Separation)

IETF ForCES aims to define a framework and associated mechanisms for standardizing the exchange of information between the logically separated functionality of the control plane (including routing protocols, traffic engineering link maintenance, admission control, and signaling) and the forwarding plane (including per-packet processing, packet forwarding, queuing, and protocol data unit (PDU) header editing). Fig. 1 shows examples of control elements (CEs), forwarding elements (FEs), and their interactions using ForCES protocol. Having standard mechanisms allows CEs and FEs to be developed by different vendors and interoperate with each other [6-9]. ForCES will enable rapid innovation in both control and forwarding planes while maintaining interoperability. Scalability is also easily provided by ForCES architecture where additional forwarding or control capacity can be added to existing network elements without the needs of big change in system architecture.

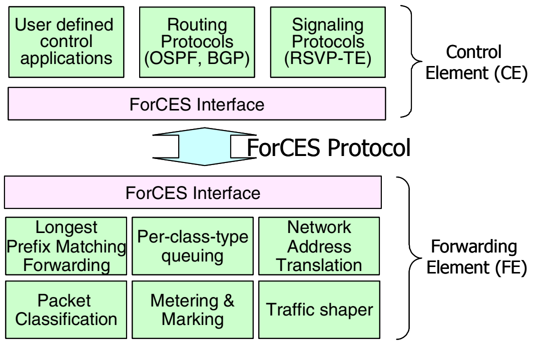


Fig. 1. Examples of control elements, forwarding elements and interactions with ForCES protocol

In ForCES architecture the physical forwarding elements may be implemented by using multiple network processors, ASICs, general purpose processors, installed on line cards, daughter boards, mezzanine, or stand-alone boxes. The control element and the forwarding element may be in close proximity (same room or small number of hops) or in very short distance (same box or single hop). In real implementations, the control elements may also be distributed on several functional modules for better performance. For example, the link status monitoring and update function in OSPF for multiple high-speed links in a large scale IP/MPLS router requires time consuming processing, and can be distributed to multiple network processors that control and manage multiple physical ports individually. Also, with distributed link status monitoring module, we can implement fast fault discovery and fast link restoration.

2.3 Distributed Control Plane with Distributed OSPF Link Status Monitoring Supported by Bidirectional Forwarding Detection (BFD)

The control plane functions, such as IP routing protocols (OSPF, IS-IS and BGP, and MPLS signaling (RSVP-TE), are generally implemented on a centralized controller for the whole IP/MPLS router/switch. For better scalability and functionality, some part of control functions may be carefully distributed to multiple functional modules which utilize high-speed multiprocessing with network processor. As an example, the periodic link status monitoring for each data link of OSPF can be distributed to each network interface module that can exchange BFD/OAM message periodically with its neighbor network interface module.

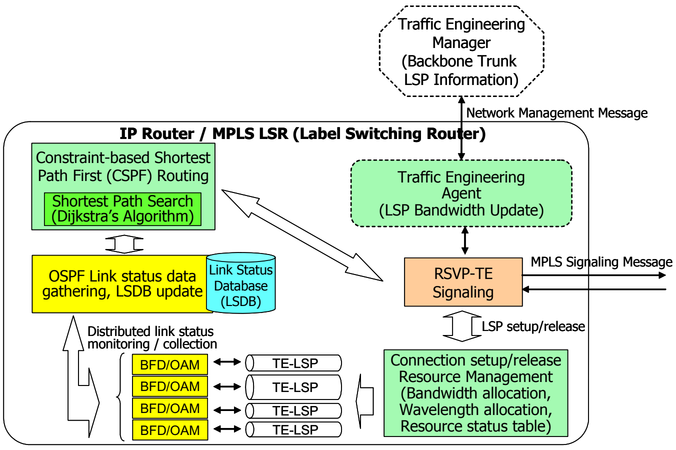

Traffic Engineering Manager (Backbone Trunk LSP Information)
Network Management Message
IP Router / MPLS LSR (Label Switching Router)
Constraint-based Shortest Path First (CSPF) Routing
Shortest Path Search (Dijkstra's Algorithm)
Traffic Engineering Agent (LSP Bandwidth Update)
OSPF Link status data gathering, LSDB update
Link Status Database (LSDB)
RSVP-TE Signaling
MPLS Signaling Message
LSP setup/release
Distributed link status monitoring / collection
BFD/OAM
BFD/OAM
BFD/OAM
BFD/OAM
TE-LSP
TE-LSP
TE-LSP
TE-LSP
Connection setup/release Resource Management (Bandwidth allocation, Wavelength allocation, Resource status table)

Fig. 2. Distributed Control Plane Architecture of IP/MPLS Router

Fig. 2 shows the functional architecture of distributed control plane, where the OSPF link status data gathering and LSDB (Link Status Database) update in a large scale IP/MPLS are distributed to multiple network interface modules that utilize high-speed packet processing and parallel processing with network processor. BFD/OAM function is implemented for each physical link or TE-LSP that is used for traffic engineering trunk. The connection setup and release function can also be partially distributed to network interface module for increased performance of control plane.

When physical layer protocol supports a well defined OAM function, such as SONET/SDH transmission system, the link status monitoring can utilize the performance measurement and fault monitoring function of the physical layer. If the physical layer protocol does not support well defined OAM functions as in Gigabit Ethernet link, however, the IP/MPLS layer protocol must implement BFD (bidirectional forwarding detection) function to detect the connectivity failure of each data link. Within the network interface module where multiple network processors might be used, the embedded processor (i.e., Xscale embedded processor in IXP2400/2800) in each network processor can execute these partial control element functions.

For fast detection of any connectivity failure in physical and logical data link protocol layer between two network interface modules, the BFD should be used. For example, in order to achieve the fault detection and recovery performance of the SONET physical layer which is limited to 50 ms, the BFD message must be periodically exchanged within every 5 ~ 10 ms, to enable the network interface module to decide logical path/link failure based on 3 consecutive BFD message losses.

3 Design of BFD/OAM for Management of DiffServ-over-MPLS Transit Network

3.1 BFD/OAM for QoS-Guaranteed DiffServ-over-MPLS Service Provisioning with Hierarchical Traffic Grooming

In order to guarantee the pre-configured QoS for DiffServ-over-MPLS, the virtual overlay network for the class-type must be continuously monitored, and the available bandwidth and edge-to-edge packet transfer delay must be continuously measured and analyzed. Fig. 3 shows the traffic grooming with hierarchical MPLS TE-LSP label stacking in DiffServ-over-MPLS virtual overlay networks, and their associated OAM functions.

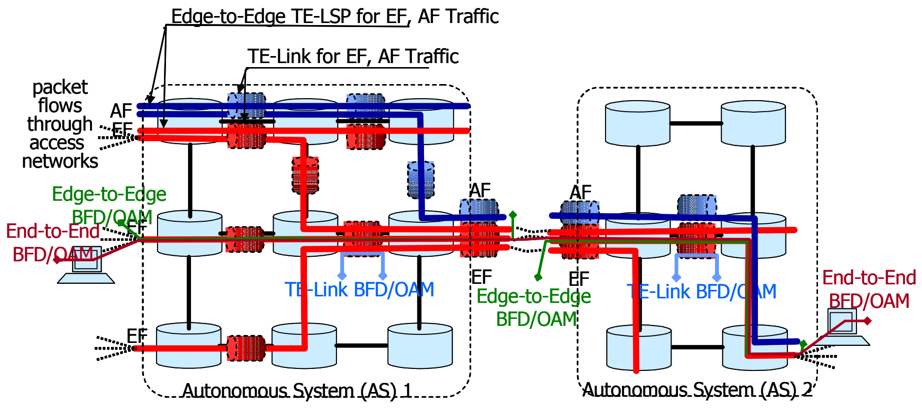


Fig. 3. Traffic grooming and associated OAM/BFD function

Each network interface module can implement the BFD function for edge-to-edge TE-LSP for each class-type, and for the aggregated TE-LSPs of a class-type between adjacent IP/MPLS routers. BFD for each TE-LSP and TE-Link will send periodic monitoring packet with time stamp to measure the packet transfer delay, jitter (delay variation), packet error rate, and packet loss rate. The distributed control function will update the link status periodically, and allow the centralized OSPF daemon to retrieve the most up-to-date link information. If there is any abnormal condition on any TE-Link or TE-LSP, the distributed control element on network processor should inform the fault to the centralized OSPF daemon immediately.

3.2 Design of ForCES Based Distributed Control Plane

Fig. 4 depicts an example of highspeed packet switch architecture with multiple control elements and forwarding elements distributed in multiple functional modules. In this architecture a centralized controller with signaling functions will control the overall routing and switching of user packet flows. The centralized controller is usually implemented as a special control module in the router/switch node, or can be implemented as a remote control node system. Partial control functions, such as link monitoring of OSPF by BFD/OAM, may be distributed at each network interface module where forwarding elements are collocated. Multiple forwarding engines are used to support multiple physical link interfaces for 1 ~ 40 Gbps rate.

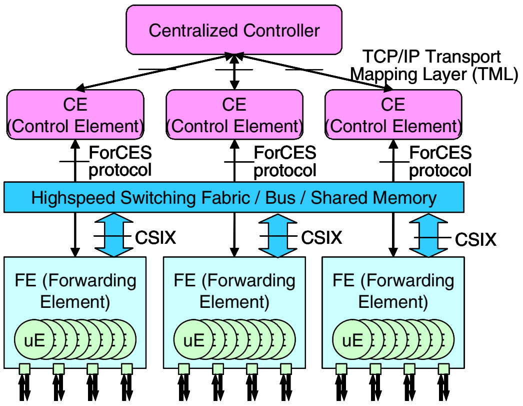


Fig. 4. Distributed control elements and forwarding elements with multiple network interface modules

One forwarding element may contain 1 ~ 2 network processors to support multiple link/port interfaces, and each forwarding element is connected to a high-speed switching fabric/bus or shared memory via CSIX (common switch interface) to support packet switching among different forwarding element modules.

The ForCES protocol provides the communication functions among CE-FE for resource discovery, establishment of associations, configuration, query and response, event notification, redirection of IP packets, and heartbeat messaging. When the partial control element is collocated with forwarding elements on a network interface module where multiple network processors are used, the communication between the partial control elements and the forwarding elements may be implemented within the same network interface module. So, the ForCES communication can be much simpler than the communication among remote systems. The communication between the centralized controller and the partially distributed control elements can be implemented with TCP/IP transport mapping layer (TML) [10].

3.3 BFD/OAM with ForCES Functional Blocks on IXDP2400

Fig. 5 depicts the BFD/OAM component block diagram on Intel IXDP2400 platform. The BFD/OAM configuration component provides interface functions, such as session creation/deletion, BFD/OAM activate/deactivate, performance analysis, fault detection and notification, and clock synchronization. BFD/OAM core component is using BFD/OAM session table that contains the detailed information of the BFD session for each TE-LSP. For each TE-LSP creation, the BFD session entry is created and the BFD/OAM activity is initialized as default *deactivated* state. In this state BFD/OAM does not transmit polling packets, but does respond to the polling packets transmitted from remote side. When the control plane activates the BFD/OAM function through ForCES protocol, the BFD/OAM core component starts periodically transmitting polling packets and receiving BFD/OAM respond packets.

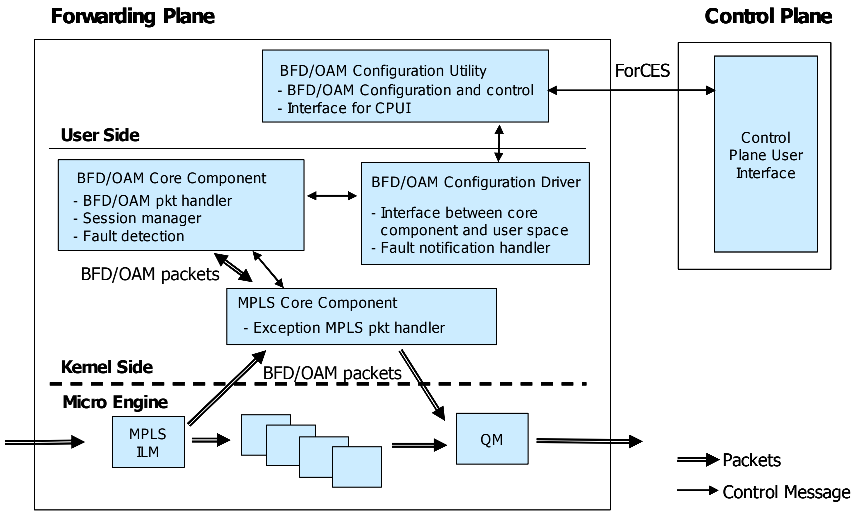


Fig. 5. BFD/OAM related functional blocks

BFD/OAM Core Component (CC) functional block diagram is shown in Fig. 6. BFD/OAM CC maintains two basic data structures: BFD/OAM session table and BFD/OAM active session list. BFD/OAM session table stores BFD related information about bi-directional link. BFD/OAM active sessions list store sessions that are currently checking their link status. BFD/OAM active sessions list entries and

BFD/OAM sessions table entries are cross-linked, to avoid search of entries and allow faster processing. MPLS CC provides couple validation message handling to allow BFD/OAM CC to know whether LSPs in the couple are valid or not. Also if any LSP participating in BFD/OAM session does not exist anymore by some reason (e.g., removed), BFD/OAM CC is informed that LSP couple is not valid anymore. BFD/OAM configuration driver registers fault notification handler in BFD/OAM CC and it is called once any link changes its status.

MPLS microblock forwards all packets containing Router Alert label to the MPLS CC as exception, which forwards them to BFD/OAM CC. BFD/OAM CC packet handler processes the packet according to usual BFD processing. If received packet is polling, response is sent, if received packet is response, timestamp of last received packet and average Round-Trip Time (RTT) is updated. Also, if the link is *DOWN*, when the response is received, its status is changed to *UP* and fault notification handler is invoked to inform control plane about link status change.

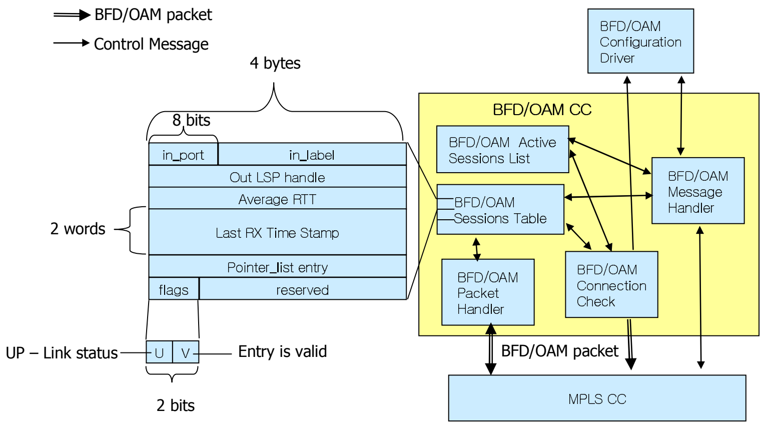


Fig. 6. BFD/OAM Core Component Functional Block Diagram

BFD/OAM core component creates a separate thread for active sessions list traversal. When the traversal starts, time is saved, and next traversal is scheduled after 5 ~ 10 ms. If the traversal takes more than 5 ~ 10 ms, next traversal is scheduled immediately after the previous is finished. Linked list entries and timestamps are accessed by one thread at the same time to avoid data corruption. During traversal BFD/OAM packets are transmitted for each active session, and also time difference between current time and last received packet timestamp is calculated. If this difference exceeds the pre-defined limit (i.e., 15 ms for BFD/OAM interval of 5 ms), connection is marked as *DOWN*, and fault notification handler is executed.

3.4 Clock Synchronization Among Distributed BFD/OAM Modules

Clock synchronization among network processors is another important issue for correct analysis of the packet delivery delay of the link or tunnel. In order to increase

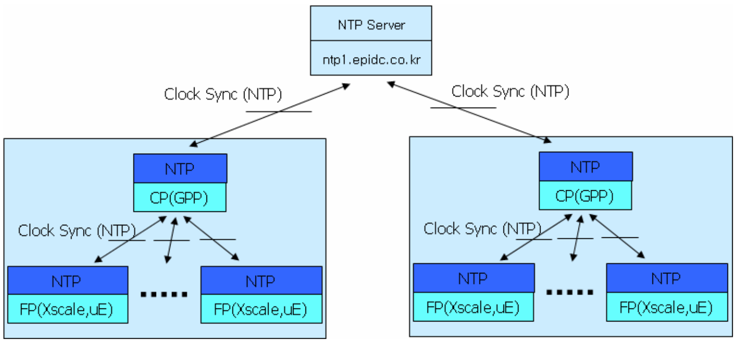


Fig. 7. NTP based clock synchronization of BFD/OAM modules

the clock precision for link failure detection, the time clock of BFD/OAM transmitter and receiver must be synchronized in micro-second order.

We enhanced the network time protocol (NTP) version 4 [18] implemented in Monta Vista Embedded Linux system on IXPD2400, to synchronize the network processors. Fig. 7 shows the NTP based clock synchronization for BFD/OAM. The system clock in each network processor is synchronized with higher precision (less than 11 us) with enhanced NTP protocol with time stamp in micro-second order.

4 Implementation and Performance Analysis of Distributed Control Plane on Intel IXDP2400 Platform

4.1 Implementation of BFD/OAM on Intel IXP2400 Network Processor

In real implementation of large scale IP/MPLS router/switches, each network interface module will include 2 ~ 10 optical ports, multiple network processors, shared memory, and optional switching fabric block with CSIX (common switch interface). The network interface modules and backbone switching module will comprise the forwarding element (FE) function. The control element (CE) will be mostly implemented on the system controller board that contains signaling protocols (RSVP-TE), routing protocols (OSPF or ISIS, BGP), and open service architecture (OSA) interface. Some part of the control element function should be distributed on each network interface module to increase the scalability and fast processing. For the scalability of control plane and packet forwarding plane, the overall system must be optimized in parallelism while minimizing the inter-module communication overhead.

We implemented the proposed distributed control plane on IXDP 2400 network processor development platform and Linux host machine. Centralized control plane function of OSPF daemon is implemented on a remote Linux host machine, and BFD and OAM functional modules are implemented on the embedded Xscale processor in IXP2400. The communication between OSPF daemon and the BFD/OAM module is implemented with TCP/IP socket, and one IXP2400/2800 network processor controls

4 ports in IXDP2400 (10 ports in IXDP2800) of 1 Gbps Gigabit Ethernet interface. For each port, a dedicated thread of embedded Linux is created which periodically sends / receives BFD/OAM packet to/from its neighbor. BFD standard defines uni-directional link status monitoring with return path. In our implementation, for efficient processing of bidirectional link status monitoring and analysis, we use piggyback mechanism in BFD/OAM packet exchange.

Fig. 8 shows the BFD/OAM packet which includes fields of discriminators, Tx and Rx intervals, minimum response Rx interval, sequence number, time stamps for delay measurements, total transmitted packet count and size for packet loss/error analysis. In current implementation, the BFD/OAM packet is sent every 5 ~ 10 msec, containing the time stamps and packet transmission statistics data. The receiving thread records the arrival time of the BFD/OAM packet, checks the packet reception statistics data from the micro engine that handles the input port, compares the transmission statistics data from the BFD/OAM packet, and replies a BFD/OAM packet with piggybacked response data.

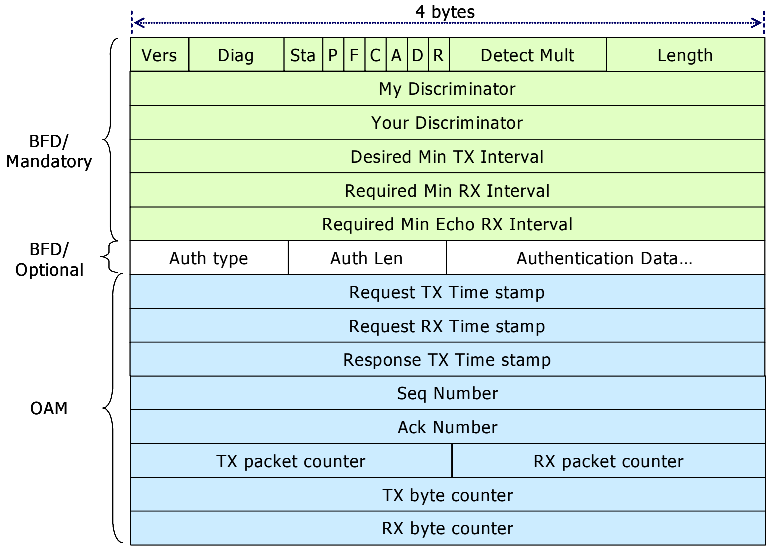


Fig. 8. BFD/OAM packet format

4.2 Analysis of Failure Detection Performance with BFD/OAM

Fig. 9 shows the interaction between the control element and forwarding element with BFD/OAM function. The control element configures the operation mode of BFD/OAM function, specifying the interval of BFD/OAM packet delivery and event notification condition. The fault restoration procedure should be implemented as an additional network management function. For faster link failure detection, the BFD/OAM packet delivery interval should be shortened.

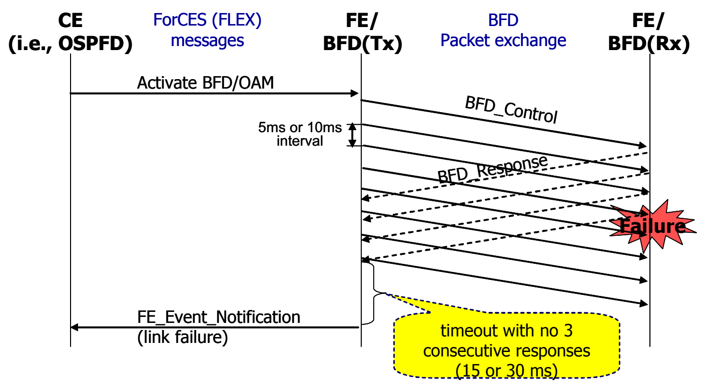


Fig. 9. Procedure of BFD packet exchange

In order to provide 50 ms link-failure restoration performance, as in SONET transmission system, we configure periodic BFD/OAM packet exchange, and if 3 consecutive BFD/OAM response packets do not arrive in expected time (i.e., 15 or 30 ms), it determines that a link failure occurred, and sends a link failure notification.

Table 1 shows the link failure detection time with BFD/OAM that has been implemented on Intel IXP2400 network processor. As shown in Fig. 5, the BFD/OAM core component periodically generate BFD/OAM packet periodically through MPLS core component that delivers the BFD/OAM packets through TE-LSP for link fault & performance management. When 3 consecutive BFD/OAM packet losses are used to indicate link failure detection, around 8 ms was taken to determine the link failure occurrence. As BFD/OAM period is reduced from 10 ms to 5 ms, the link failure detection time can be shortened from 38.029 ms to 23.095 ms.

Table 2 shows the overhead of BFD/OAM for TE-LSP. As the BFD/OAM interval is shortened, the transmission rate of BFD/OAM increases, and thus the overhead for the TE-LSP. When the TE-LSP transmission rate is more than 10 Mbps, however, the overhead of BFD/OAM with 5 ms interval is less than 1 %. Another consideration is the processing time for BFD/OAM by the network processor. In Intel IXP2400 network processor, the maximum number of active sessions is limited by 77 and 38 because of the processing speed limit of Xscale processor at the BFD/OAM interval of 10 ms and 5 ms, respectively.

Table 1. Link failure detection time with BFD/OAM

BFD/OAM period	Link failure detection time	Remark
10 ms	38.029 ms	Excluding propagation delay
5 ms	23.095 ms	Excluding propagation delay

Table 2. BFD/OAM overhead

TE-LSP Transmission Rate	Polling interval	
	10 ms	5 ms
1 Mbps	5.4400%	10.8800%
10 Mbps	0.5440%	1.0880%
100 Mbps	0.0544%	0.1088%
622 Mbps	0.0087%	0.0175%
1 Gbps	0.0054%	0.0109%
2.4 Gbps	0.0023%	0.0045%
10 Gbps	0.0005%	0.0011%
Maximum number of active sessions (IXP2400)	77	38

5 Conclusion

In this paper, we designed and implemented the management functions of DiffServ-over-MPLS transit network with BFD/OAM in ForCES architecture for QoS-guaranteed broadband realtime multimedia service provisioning. The proposed BFD and ForCES functions are implemented with Intel IXP 2400 network processor, where BFD/OAM packets for MPLS TE-LSP are exchanged every 5 ms or 10 ms for performance measurements and link failure detection. The operations of BFD/OAM-based link failure detection and performance measurement are controlled via distributed control plane with ForCES architecture for large scale IP/MPLS router using multiple network processors in each network interface card (NIC).

We analyzed the processing overhead and maximum number of active BFD/OAM session that can be configured on IXP2400 network processor. With the BFD/OAM functions with less than 5 ms interval, we could implement protocol-independent fast link failure detection within 23 ms (excluding the propagation delay) without sophisticated link failure detection in physical layer. The proposed BFD/OAM function also provides performance measurement of delay, jitter, packet loss/error for TE-LSPs in DiffServ-over-MPLS virtual overlay transit networks. The measure QoS parameters of TE-LSPs are used in the constraint-based shortest path first routing for QoS-guaranteed multimedia service provisioning across multiple domain networks.

References

1. Young-Tak Kim, Hae-Sun Kim, and Hyun-Ho Shin.: Session and Connection Management for QoS-guaranteed Multimedia Service Provisioning on IP/MPLS Networks. Proceedings of ICCSA2005 (LNCS 3481). (2005) 157 ~ 168
2. IETF Bidirectional Forwarding Detection (bfd) working group.: http://www.ietf.org/html.charters/bfd-charter. html
3. D. Katz, et. al.: Bidirectional Forwarding Detection. IETF Internet Draft (2005)
4. D. Katz, et. al.: BFD for IPv4 and IPv6 (Single Hop). IETF Internet Draft (2005)
5. D. Katz, et. al.: BFD for Multihop Paths. IETF Internet Draft (2005)

6. L. Yang, et. al.: ForCES Architecture Framework. IETF RFC 3746 (2004)
7. A. Doria.: ForCES Protocol Specification. IETF Draft, draft-ietf-forces-protocol-01.txt (2004)
8. A. Audu, et. al.: Forwarding and Control Element Separation IP Transport Mapping Layer. IETF Draft, draft-audu-forces-iptml-00 (2004)
9. Furquan Ansari, et. al.: ForCES Intra-NE Topology Discovery. IETF Draft, draft-ansari-forces-discovery-01.txt (2004)
10. Hormuzd Khosravi, et. al.: TCP/IP based TML (Transport Mapping Layer) for ForCES protocol. IETF Draft (2004)
11. Raul Aggarwal, et. al.: BFD for MPLS LSPs. IETF Internet Draft (2005)
12. Douglas E. Comer.: Network Systems Design using Network Processors. Prentice Hall (2004)
13. Bill Carlson.: Intel Internet Exchange Architecture and Applications. Intel Press (2003)
14. Erik J. Johnson and Aaron R. Kunze.: IXP2400/2800 Programming. Intel Press (2003)
15. Intel IXP2400/IXP2800 Network Processors – Microengine C Language Support Reference Manual (2003)
16. Intel Internet Exchange Architecture Software Development Kit – Software Framework Installation Guide, Intel (2004)
17. Intel Control Plane – Platform Development Kit, Intel (2004)
18. Network Time Protocol (NTP) Distribution (2005)

Detecting Bottleneck in *n*-Tier IT Applications Through Analysis

Gueyoung Jung[1], Galen Swint[1], Jason Parekh[1], Calton Pu[1], and Akhil Sahai[2]

[1] CERCS, Georgia Institute of Technology
801 Atlantic Drive, Atlanta, GA 30332
{gueyoung.jung, galen.swint, jason.parekh, calton}@cc.gatech.edu
[2] HP Laboratories
Palo-Alto, CA
akhil.sahai@hp.com

Abstract. As the complexity of large-scale enterprise applications increases, providing performance verification through staging becomes an important part of reducing business risks associated with violating sophisticated service-level agreement (SLA). Currently, performance verification during the staging process is accomplished through either an expensive, cumbersome manual approach or ad hoc automation. This paper describes an automation approach as part of the Elba project supporting monitoring and performance analysis of distributed multi-tiered applications that helps in bottleneck detection. We use machine-learning to determine service-level objectives (SLOs) satisfaction and locate bottlenecks in candidate deployment scenarios. We evaluate our tools with TPC-W, an on-line bookstore, and RUBiS, an on-line auction site.

Keywords: Bottleneck detection, *n*-tier application, Decision tree, SLOs, Elba.

1 Introduction

The increasing complexity of enterprise applications has emphasized the importance of verifying and validating the configuration performance prior to production use. While functional properties are typically verified during system integration and testing, performance as specified in SLOs of SLA is verified and validated by a pre-production process referred to as *staging*. Since a failure to fulfill SLA requirements results in business losses, staging has the critical role of verifying and validating the deployment plan to cover a wide range of system configurations and workloads. Current staging processes have been largely manual, augmented occasionally with ad hoc automation scripts, and these processes have become increasingly error-prone and costly in terms of time and effort. To reduce costs and increase the coverage of staging, the Elba project seeks to automate staging and tuning for *n*-tier applications in a distributed environment [4]. Automated staging and tuning uses high-level requirement specifications and translates them into both staging deployment and workload parameter settings which are then used for execution in a staging environment. Monitors first collect performance data, followed by analysis which automatically identifies performance deficiencies. Generating and analyzing performance data to uncover SLOs satisfaction and performance bottlenecks are significant challenges to realizing tiered, self-tuning applications.

Our prior work in the Elba project, represented in Figure 1, reported on the challenges of automatically mapping design specifications into deployment tool specifications for production and provided a solution using a code generation and translation tool [4][5].

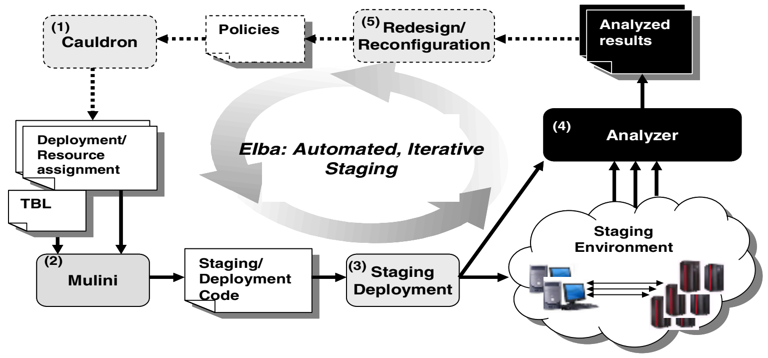


Fig. 1[1]**.** Staging in Elba is an automated and iterative process. (1) Cauldron converts the policy documents into resource and deployment assignments. (2) Mulini re-maps resource assignments and application staging test guideline (TBL) to generate three types of code: instrumented application code, deployment code, and monitoring/analysis code. (3) A deployment tool installs and configures the application and then executes it. (4) Monitoring data is fed into analysis tools, and (5) the result of the analysis is handed to an engine to generate recommendations for policy changes. In this paper, we focus on (4), highlighted by black boxes, for the automated monitoring and analysis using generated code from Mulini code generator. Note that dashed boxes are on progress.

This paper presents our work on collecting data and analyzing bottlenecks for a significant number of performance metrics. It helps answers two questions:

- *Does the application configuration meet performance requirements?* This question is answered by observing metrics that correspond to policy objectives in SLOs.
- *If the requirements are not met, then where in the configuration is the bottleneck?* This question should be answered by examining metrics data establishing, first, what metrics are relevant, and second, which metrics best represent the bottleneck.

Neither task is trivial, but both are valuable for staging and production. Of course, over-provisioned systems can meet SLOs, but this entails additional capital outlays, maintenance, and sometimes over-engineering of the software itself [8]. Systematic, automated staging mitigates the risks of under- and over-provisioning and can provide valuable application behavior information applicable to the production application.

The contribution of this paper is an approach to support automated monitoring, analysis, and reporting by applying machine-learning in the context of staging. This automated approach will assist service providers in answering the previous two questions while preventing resource wastage through over-provisioning. With Mulini code generator improved from [4], our approach uses policy documents to generate metrics monitoring and performance analysis code and hooks into a machine-learning tool for

[1] We have slightly improved the figure of Elba used in [4] and [12].

automated bottleneck detection. We compared different classifiers and decided that the decision tree classifier (J48) was more robust in detecting bottlenecks [12]. In this article, we evaluate the accuracy of our bottleneck detection approach by analyzing two well-known benchmarking applications[2] that have differing bottleneck profiles, TPC-W and RUBiS.

The remainder of this paper is organized as follows. Section 2 presents the challenges and our approach to providing analysis support in the Elba project. Section 3 describes the evaluation environment, and Section 4 presents evaluation results for TPC-W and RUBiS. Section 5 discusses related work, and Section 6 presents our conclusions.

2 Automated Staging and Analysis

2.1 Challenges

For a distributed *n*-tier application, staging is an important, complex task that entails repeated tests over an extended period of time; the system and configuration are refined until they meet performance requirements. If performed early in the application development cycle, staging can provide crucial feedback that helps steer application development by identifying bugs, performance shortfalls, "hotspots," and resource waste. Increasing application complexity makes staging worth automating to enable faster, earlier testing.

Staging may share some tools and techniques with production, but three important factors differentiate application staging from production. First, the hardware available in a staging environment mirrors but may not replicate exactly the production environment. Perfect duplication would provide higher application assurance once staged, but involves high costs in terms of initial expenditures and ongoing maintenance. To best utilize an approximate environment that minimizes the costs requires staging the application multiple times to establish predictive performance trends. Each staging iteration tests one particular application configuration and may involve multiple staging trials under varying staging parameters. A second differentiator is that applications may require additional fine-grained implementation to ascertain bottlenecks accurately which must be removed from the production code. Finally, staging requires the generation of synthetic workloads that stress the application similarly to production environments in a limited time period.

Automated analysis adds challenges to the staging process. First, automated analysis entails the orchestration of several tasks and may drive multiple executions with slightly different staging parameterizations. Second, automated analysis requires system and application instrumentation derived from performance requirements to record metrics data. Third, it requires the construction of an analysis, decision, and detection process which can answer the two questions presented in the introduction.

Automated analysis tools must translate policy-level documents into functional artifacts that become part of the staging process. Service-level indicators (SLIs) are

[2] In this paper, they are used as exemplar distributed multi-tiered e-commerce applications with defined metrics rather than as benchmarks; our results can not be used for performance comparisons outside this paper.

obtained from the SLA and its components, SLOs; these are translated into metrics and staging parameters. Furthermore, administrator/operator policies that may govern aspects such as acceptable resource usage must also be mapped into metrics. Test-specific information, such as machine locations, testing times, and workload, must be incorporated. Once recorded, a custom analysis engine automatically processes the data and compares its results against performance goals set forth in policy specifications.

Even automating bottleneck detection from gathered data requires the recognition and resolution of several problems. First, a single trial may not provide enough information to determine bottlenecks – a metric may appear "maxed" out even though it really reflects normal operating levels. In such cases, several trials of varying workloads are required to establish operating baselines and trends. Interactions between metrics can also make bottleneck detection difficult. For example, CPU usage and network throughput may trend in parallel, but only the CPU is the bottleneck. Finally, bottleneck detection requires sorting through copious metrics data. The total number of metrics varies with the number of both hardware and software components in the system, and they can be categorized generally as either application-level (e.g., the number of threads, the number of database connections, and elapsed query time) or system-level (e.g., CPU and memory utilization) metrics.

To detect bottlenecks and sort through the myriad metrics produced during monitoring, we employ an automated classifier. The input to the classifier is the first derivative of the metrics, since we are interested in trends. It is first trained by inputting the metrics with the result of a SLO-evaluator, a tool generated for deciding the violation of the SLOs. The output of the classifier is metrics whose derivatives correlate strongly to SLOs violation. From these identified metrics, we discover the bottlenecks of the system.

2.2 Automating Monitoring and Analysis

Our approach to automated staging and analysis occurs within the context of the Elba project [4]. The project goal is to first realize iterative staging to determine the inadequacies of application performance, then evaluate the results, and finally enable automated tuning of the system to meet the expected performance objectives. In particular, to integrate our automation for monitoring and analysis, we extended Elba's Mulini code generator which employs XML/XSLT techniques with Aspect Oriented Programming (AOP) paradigm to create the necessary code for monitoring and analysis, including the instrumentation of source code for application-level metrics, and for the generation of the analysis code. Interested readers for code generation and Mulini can refer to [4][9] for more details. The metrics data can then be fed to a machine-learning tool to identify performance bottlenecks. The Analysis addresses the two questions posed in the introduction, namely,

Does the application configuration meet its performance requirements?
This must be answered for each trial. Mulini generates an SLO-evaluator with policy-specific code that computes the individual satisfaction of the component SLOs. The SLO-evaluator uses data collected by the synthetic workload generators and computes application-specific throughput and average response time. Once the SLOs satisfaction is determined, overall SLA satisfaction can be determined.

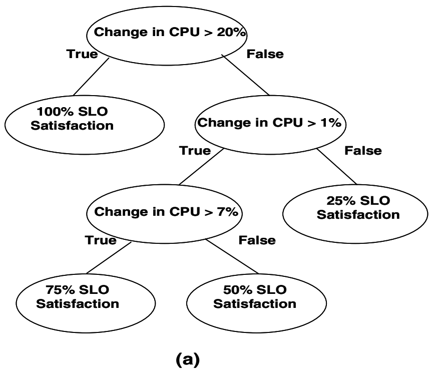
If the requirements are not met, then where in configuration is the bottleneck? This question is answered with aggregated data from multiple trials. If SLA is not met, the tools begin a three-step bottleneck detection process to correlate performance shortfalls and metrics. The detection process requires performance data from a series of trials. The first trial subjects the application to a low synthetic workload, and each subsequent trial increments the workload until consecutive trials fail the SLA. For example, a trial for a retail store application may begin with 10 concurrent simulated users, and then in each subsequent trial the number of concurrent simulated users increases by 10 over the previous trial until the SLA is violated for 70, 80, and 90 users.

In subsequent analysis, the first step is to determine the bottleneck tier. For each tier, the average duration spent by each service request is computed, and we identify the bottleneck tier as tier with the fastest growing duration (change in duration divided by the change in synthetic workload). The second step is to select, from the metrics of the bottleneck tier, the highly utilized metrics as candidate indicators. The assumption is that high utilization of a resource implies high demand from the application and a potential bottleneck. This also helps distinguish between highly-correlated metrics, such as bandwidth and CPU usage. To be considered a candidate indicator, a metric must either surpass 90% utilization or some threshold value as specified by a policy document, heuristic, or system administrator. The third step in bottleneck detection is to discover the metrics indicating bottlenecks using the aggregated performance data from all trials. For our applications and metrics, we have found that using the change in a metric from trial to trial provides a reliable indicator for correlating a metric to SLOs violation. The change, effectively a first derivate of the metric, will drop from some positive factor (utilization increases) towards around zero (utilization constant) when the underlying resource is fully utilized. In comparison, a non-bottleneck metric can continue to increase – constant growth does not correlate a change in SLOs satisfaction. In other words, our bottleneck detection searches out the metric that best correlates to reduced SLOs satisfaction (i.e., greater SLOs violation).

For the third step mentioned above, we apply machine-learning to form a decision tree where tree nodes embody if/then decisions based on growth in a metric (the delta metric value) and whose leaves embody overall SLOs satisfaction. After training, the set of nodes traced from a leaf (SLOs satisfaction) to the root will be a set of inequalities that is able to distinguish the leaf prediction attribute from the other prediction attributes. We categorize the satisfaction levels as quintiles since the decision tree classifier must have nominal types for prediction attributes. Five categories balance enough categories to allow correlations with each category to still collect multiple trials from the training set. By inspecting the generated decision tree, the bottleneck detection process is able to find the metric that was identified to have the highest correlation to SLOs. In situations where the decision tree consists of multiple metrics, the metric that appears most often in the tree will be selected as the highest potential bottleneck. To illustrate the bottleneck detection process, we present a sample scenario where CPU is the bottleneck metric and memory is shown to be high but not considered a bottleneck as it is cached data that accounts for most of the memory (the cached data will be replaced if an application requires additional memory). In Figure 2 (b), the utilization for both CPU and memory is shown along with the SLOs satisfaction at each workload trial. The first step for bottleneck detection is taking the difference of each metric's utilization across the change in workload trials. The trends resulting from this are shown in Figure 2 (c) where the delta CPU utilization is somewhat linear until flattening out at 0% (in which case its utilization reaches 100%), and the delta memory utilization remains mostly constant around 0%-1%. By feeding this data to a decision tree classifier, we obtain a sample tree similar to Figure 2 (a). In this case, the CPU metric was chosen at each node as its delta is most correlated to the different SLOs satisfaction categories. Memory was not chosen as it is not possible to use the delta memory utilization to differentiate each of the SLOs satisfaction categories.

3 Evaluation Environment

We evaluated the described automation approach by using TPC-W, an on-line bookstore application for a transactional web-based e-commerce benchmark [2][3], and RUBiS, e-commerce application implementing the core features of an online auction site [2]. These applications have differing performance characteristics, as described in [2]. In both, customer interaction is simulated by remote simulated browsers that send and receive HTTP messages. Each simulated browser starts from a home interaction and executes another interaction after "thinking" for a random period of time. The visitation path is governed by a chosen transition matrix which encodes probabilities for visiting the next page according to current visiting page. For our tests, we chose shopping transition and bidding transition models for TPC-W and RUBiS, respectively, since these are the most representatives of the workload of these applications as described in [2][4].



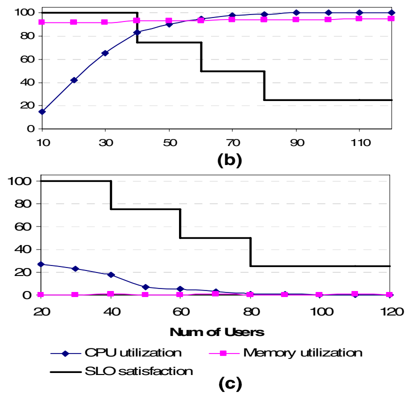


Fig. 2. (a) decision tree, (b) metric utilization, and (c) delta graph

In our evaluation, we used two software architectures common in the e-commerce domain: Java servlets for TPC-W and Enterprise Java Beans for RUBiS. A minimum configuration of the TPC-W in our evaluation consists of a web server (Apache), a servlet engine (Apache Tomcat), and a database server (MySQL) each running on a dedicated host; a minimal TPC-W installation requires three machines. A minimal RUBiS configuration comprises a web server (Apache), a servlet engine (Tomcat), an EJB server (JOnAS), and a database server (MySQL). The servlet engine and EJB server (the application tier) share a single machine. Beginning with these minimal configurations, we iterate through more complex configurations by employing higher-performance machines or adding new machines to each bottleneck tier until a configuration satisfies the given SLA.

We employ two classes of hardware in our evaluation. A low-end machine, L, is a Pentium III 800MHz dual-processor with 512 MB memory, and a high-end machine, H, is a Xeon 2.8GHz dual-processor with 4GB memory. A configuration may combine these two classes of hardware. For instance, the L/2H/L configuration represents one low-end machine for a web server, two high-end machines as application servers, and one low-end machine for a database server. All machines are connected through 100 Mbps Ethernet. For basic cost accounting of the configurations, we assign low-end machines a cost of $500 and high-end machines a cost of $3500.

4 Evaluation Results

4.1 Automated Analysis for TPC-W

Consistent with [2], the SLO-evaluator indicates that the L/L/L configuration for TPC-W fails the SLA of 10.7 WIPS with average response times less than 500 ms at staging with 150 concurrent simulated users. This triggers the process of the automated bottleneck detection with aggregated monitoring results of the L/L/L configuration, which is adjusted, and then re-staged iteratively until a configuration satisfies the SLA.

From the first step of the automated bottleneck detection process, the automated bottleneck detection identifies the database server tier as the bottleneck tier. Figure 3 shows the results of application-level monitoring in the L/L/L configuration of TPC-W. Mulini weaves monitoring code with the TPC-W application source code to record response times elapsed in database queries and execution for the presentation and business logic of "BestSeller" interaction, a representative TPC-W interaction. This figure shows that the duration of the database tier grows fastest. That is, the database server tier dominates the overall response time of the interaction. In fact, the average duration for executing database queries is about 89s while the average duration both for executing presentation and business logic and for executing requests forwarding at web server are about 1.3s when the synthetic workload generators run 150 concurrent simulated users. Once the database server tier is identified as a bottleneck, the bottleneck detection proceeds to the thresholding step, which focuses on metrics that indicate high-resource utilization.

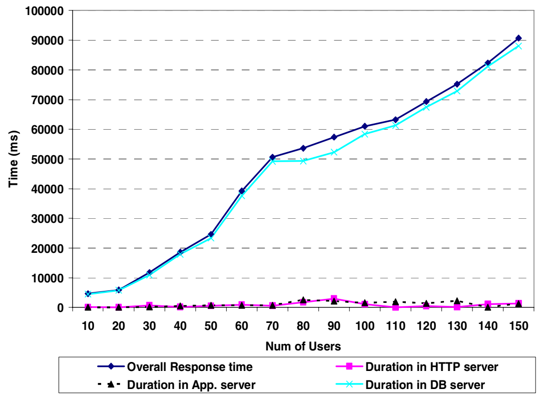
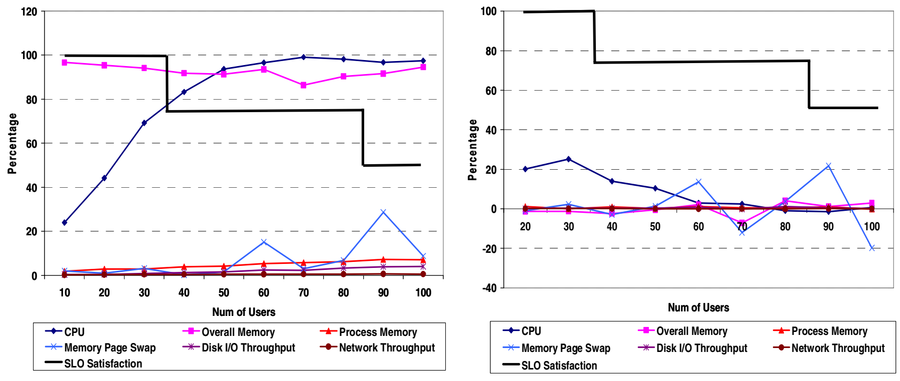


Fig. 3. Average response time and duration in each tier of BestSeller interaction

Figure 4 (a) displays each metric as a percentage of its maximum capability. Any metric not reaching 90% utilization is automatically dropped from bottleneck consideration. We can see from the figure that the metrics reaching the 90% threshold are the CPU and overall memory usage.

The final step of bottleneck detection is training the J48 decision tree classifier (WEKA toolkit's implementation of the C4.5 decision tree [6]) to locate metrics that most influence the SLOs satisfaction. Our classifier is trained with the nine derivative metrics values and the first order derivatives of the metrics to identify trends rather than the values of metrics to SLOs satisfaction. In our experience using only metric values can lead to false conclusions about which metrics are the real bottlenecks as is illustrated in this case by overall memory utilization. In Figure 4 (a), we see that the overall memory usage value is about 98% under a load of 100 concurrent simulated users. Note that the memory usage of database processes is very low. If we turn our attention to the derivatives of the metrics in Figure 4 (b), our inquiries are guided a different direction. The trend of the memory usage derivative is nearly constant. In this case, it turns out that memory is not the bottleneck because the high utilization



(a) (b)

Fig. 4. For the TPC-W database tier, (a) metric values and (b) their derivatives

stems from OS caching. Taking the derivative screens out linearly increasing metrics, which correlate but have adequate headroom for growth. The metric with the highest correlation to SLOs satisfaction is the CPU usage since the inequality "change in CPU usage > 20" can be used to differentiate the 100% SLOs satisfaction. The decision tree further differentiates the 75% SLOs satisfaction with the inequality "change in CPU usage > 2 and change in CPU usage < 15".

Since the CPU of the database server tier limits performance, we set Elba to first increase the number of low-end database server machines. This approach is much cheaper than the approach employing a few high-end machines in terms of configuration cost. The results of several iterations are shown in Figure 5. We see that only the L/L/H2L configuration (cost $5500), in which we use one high-end and two low-end machines as database servers, and the more-costly L/L/2H (cost $8000) configuration satisfy the given SLOs (configurations arranged by increasing cost). To show that neither the web server nor the application server is the bottleneck, we set Elba to conduct

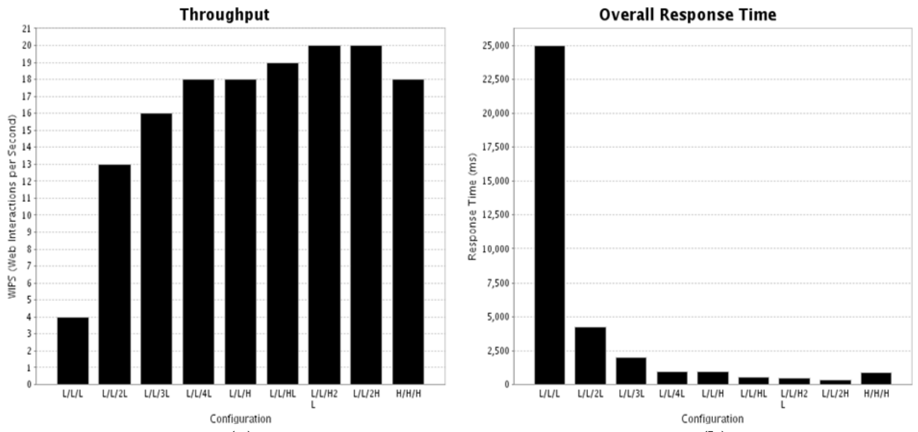


Fig. 5. TPC-W iterative staging results (a) WIPS and (b) overall average response time

extra staging with the H/H/H configuration. Fig. 5 shows that the performance results of both the L/L/H and the H/H/H configurations are almost identical in terms of WIPS and overall average response time even though we use high-end machines for both the web and the application servers. Therefore, H/H/H configuration is discovered as an over-provisioning.

Figure 6 breaks performance into per-interaction SLOs which must meet a 90% SLOs satisfaction level. Configurations that employ only cheap machines like L/L/3L cannot meet the SLOs in most interactions of the TPC-W. Using a single high-end as database server, L/L/H, also fails the SLOs for “BestSeller” and “BuyConf”. The L/L/H2L configuration narrowly meets these SLOs, and L/L/2H is clearly sufficient. From this staging result, the service provider can choose either a configuration at lower cost with less growth potential (i.e., L/L/H2L) or higher cost with high growth potential (i.e., L/L/2H).

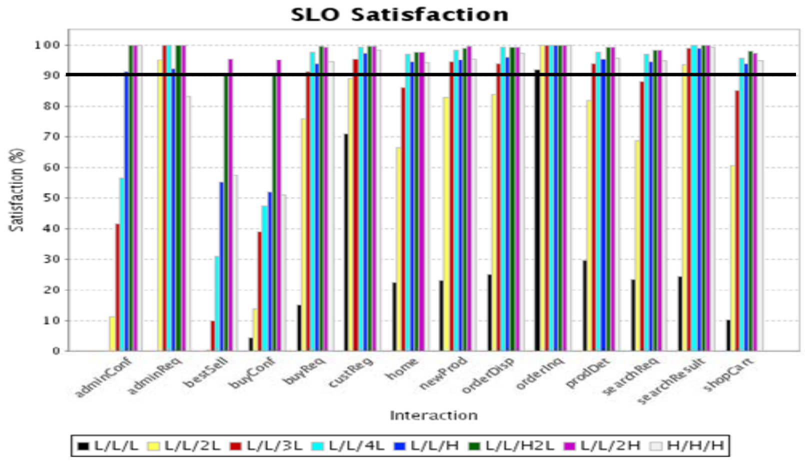


Fig. 6. TPC-W, per-interaction 90% SLOs satisfaction

4.2 Automated Analysis for RUBiS

Our SLO-evaluator indicates the L/L/L fails the target SLA of 25.7 WIPS and overall average response time of less than 500 ms under a load of 360 concurrent simulated users. This triggers the process of the automated bottleneck detection just as we have done with the TPC-W evaluation.

In the first step with "SearchItemsInCategory" interaction, we found that the application server tier dominates the overall response time of the interaction. In fact, the average duration for executing presentation and business logic (i.e., time spent in servlets and EJBs) is about 28s while the average duration in database server tier for executing database queries is about 30ms, and web server tier for forwarding requests and responses 104ms with 360 concurrent users.

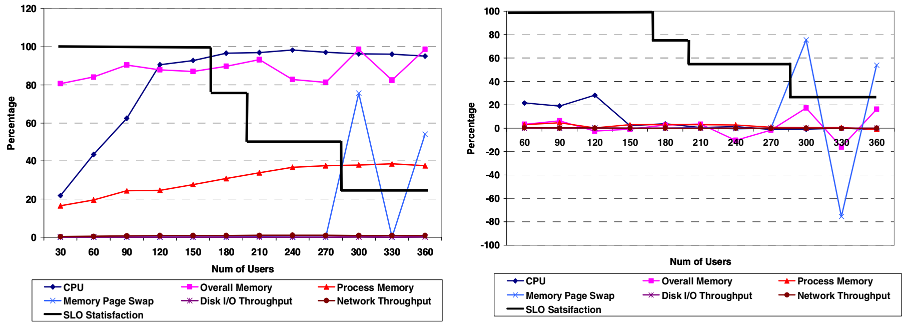


Fig. 7. For RUBiS app server tier, (a) metric values and (b) derivative values

Figure 7 (a) displays that the only metrics reaching the 90% threshold are the CPU and overall memory usage. In the final step, the decision tree classifier is trained using eleven instances. In Figure 7 (a), we see that the overall memory usage value is about 98% under a load of 360 concurrent users. However, its trend linearly increases. In Figure 7 (b), the trend of the memory usage derivative is first nearly constant and then erratic. Taking the derivative screens out the jittery metrics, which have no correlation, and linearly increasing metrics, which correlate but have adequate headroom for growth.

The metric with the highest correlation to the SLOs satisfaction is CPU usage since the inequality "change in CPU usage > 4" can be used to differentiate the 100% SLOs satisfaction. The decision tree would be able to further differentiate SLOs satisfaction with a more fine-grained distinction since the CPU reaches its peak of near-100% utilization and the derivatives approach much smaller values. The inequality "change in CPU < 1.5 and change in CPU > -1" distinguishes the 50% SLOs satisfaction.

5 Related Work

Argo/MTE [1] uses automation and code generation via XSLT to evaluate middleware implementations. The Weevil framework supports the management of testing in widely distributed systems again using a generative programming approach [11]. Weevil's focus has been on automating deployment and workload generation for applications utilizing overlay networks. Our work targets the enterprise *n*-tier IT environment and applications and emphasizes the re-use of existing policy-level specifications for automation of both performance testing and bottleneck identification with machine-learning technique.

Te-Kai et al. [10] have provided a capacity sizing tool to recommend cost-effective hardware configuration for integrated business processes; their tool is tailored to the WebSphere InterChange server. It assumes a prototype of the system is not available for system staging. Instead, it relies on similar previously benchmarked systems to predict capacity. The Elba project is geared towards staging an application that will be deployed to a production without pre-existing performance data. Our approach for bottleneck detection shares similarities with [7], but their work targets production systems to forecast problems; our work intervenes during application design to locate candidate bottleneck points, and our system also emphasizes automation support for testing alternative designs.

6 Conclusion

With the increasing complexity of large-scale enterprise applications, effective staging can ensure the SLA performance goals of complex application configurations. The goal of the Elba project is to automate iterative staging. The main contribution of this paper is the automated monitoring and performance analysis of large-scale applications through a decision tree approach for bottleneck detection assisted by code generation techniques. From declarative specifications of distributed *n*-tier applications, we generated the code to collect, process, and analyze performance data (e.g., SLOs satisfaction levels) to locate performance bottlenecks in configurations being staged.

Our evaluation results of TPC-W and RUBiS demonstrated the feasibility and effectiveness of automating the monitoring and performance analysis in the staging process. By generating and running various configurations, our tools analyzed the SLOs satisfaction levels, found potential bottlenecks, and guided the reconfiguration process towards the lowest cost solution. The analysis tool utilized simple machine-learning techniques to classify the resource consumption metrics and find potential bottlenecks.

References

[1] Cai, Y., Grundy, J., and Hosking, J.: Experiences Integrating and Scaling a Performance Test Bed Generator with an Open Source CASE Tool. Int. Conf. on Automated Software Engineering, Linz, Austria, Nov. 2004.

[2] Cecchet, E., Chanda, A., Elnikety, S., Marguerite, J., and Zwaenepoel, W.: Performance Comparison of Middleware Architectures for Generating Dynamic Web Content. Int. Middleware Conf., Rio de Janeiro, Brazil, June 2003.

[3] García, D., and García, J.: TPC-W E-Commerce Benchmark Evaluation, *IEEE Computer*, Feb. 2003.

[4] Swint, S. G., Jung, G., Pu, C., and Sahai, A.: Automated Staging for Built-to-Order Application Systems. Network Operations and Management Symposium, Vancouver, Canada, April 2006.

[5] Sahai, A., Pu, C., Jung, G., Wu, Q., Yan, W., and Swint, S. G.: Towards Automated Deployment of Built-to-Order Systems, Distributed Systems; Operation and Management, Barcelona, Spain, Oct. 2005.

[6] WEKA distribution. http://www.cs.waikato.ac.nz/ml/weka.

[7] Cohen, I., Goldszmidt, M., Kelly, T., Symons, J., and Chase, J.: Correlating Instrumentation Data to System States: A building block for automated diagnosis and control. Operating System Design and Implementation, San Francisco, CA, USA, Dec. 2004.

[8] Sauvé, J., Marques, F., Moura, A., Sampaio, M., Jornada, J., and Radziuk, E.: SLA Design from a Business Perspective, Distributed Systems: Operation and Management, Barcelona, Spain, Oct. 2005.

[9] Swint, S. G., Pu, C., Consel, C., Jung, G., Sahai, A., Yan, W., Koh, Y., and Wu, Q.: Clearwater - Extensible, Flexible, Modular Code Generation. Int. Conf. on Automated Software Engineering, Long Beach, CA, USA, Nov. 2005.

[10] Te-Kai, L., Hui, S., and Kumaran, S.: A capacity sizing tool for a business process integration, Int. Middleware Conf., Toronto, Ontario, Canada, Oct. 2004.

[11] Wang, Y., Rutherford, M., Carzaniga, A., Wolf, A.: Automating Experimentation on Distributed Testbeds, Int. Conf. on Automated Software Engineering, Long Beach, CA, USA, Nov. 2005.

[12] Parekh, J., Jung, G., Swint, S, G., Pu, C., and Sahai, A.: Comparison of Performance Analysis Approaches for Bottleneck Detection in Multi-Tier Enterprise Applications, Int. Workshop on Quality of Service, Yale University, New Haven, CT, USA, June, 2006.

Fast Extraction of Adaptive Change Point Based Patterns for Problem Resolution in Enterprise Systems

Manoj K. Agarwal, Narendran Sachindran, Manish Gupta, and Vijay Mann

IBM India Research Labs, Block 1, IIT campus,
Hauz Khas, New Delhi - 110016, India
{manojkag, nsachind, gmanish, vijamann}@in.ibm.com

Abstract. Enterprise middleware systems typically consist of a large cluster of machines with stringent performance requirements. Hence, when a performance problem occurs in such environments, it is critical that the health monitoring software identifies the root cause with *minimal* delay. A technique commonly used for isolating root causes is rule definition, which involves specifying combinations of events that cause particular problems. However, such predefined rules (or problem signatures) tend to be inflexible, and crucially depend on domain experts for their definition. We present in this paper a method that automatically generates change point based problem signatures using administrator feedback, thereby removing the dependence on domain experts. The problem signatures generated by our method are flexible, in that they do not require exact matches for triggering, and adapt as more information becomes available. Unlike traditional data mining techniques, where one requires a large number of problem instances to extract meaningful patterns, our method requires few fault instances to learn problem signatures. We demonstrate the efficacy of our approach by learning problem signatures for five common problems that occur in enterprise systems and reliably recognizing these problems with a small number of learning instances.

Keywords: fault localization, patterns, problem signatures, change point detection, adaptive learning.

1 Introduction

Modern enterprise systems are often required to provide services based on service level agreement (SLA) specifications at minimum cost. SLA breaches typically result in a significant penalty. Performance problems in these systems usually manifest themselves as high response times, low throughput, or a high rejection rate of requests. However, the root cause of these problems may be due to subtle reasons hidden in the complex stack of this execution environment. For example, badly written application code may cause an application to hang. Network problems like non availability of a connection between an application server and a database server can cause critical transactions to fail. Backup processes on a machine could cause performance degradation of servers running on that machine. Further, various components in such systems could have inter-dependencies which may be temporal or non-deterministic as they may change with changes in topology, application or workload. This further complicates root cause localization.

A commonly used event correlation technique for localizing the root cause of performance problems is rule definition [4]. In rule definition, all possible root causes are represented by rules specified as condition-action pairs. Conditions are typically specified as logical combinations of events, and are defined by domain experts. A rule is satisfied when a combination of events raised by the management system exactly matches the rule condition. Rule based systems while popular, suffer from two major drawbacks. First, they need domain experts to define rules. Second, rules are inflexible - they require exact matches and do not adapt as the environment changes.

Automatic learning of rules has been studied earlier by Hellerstein et al. [1]. They discover patterns using association rule mining based techniques [14]. They observe that when a fault occurs, it is usually accompanied by a burst of events. Additionally, each fault is usually associated with an event pattern. To corroborate these findings, we performed experiments on a multi-tier application running in a cluster. We employed change point based monitoring of performance metrics to generate alarms. The experiments consisted of several repetitions of different faults and resulted in the following observations:

- Certain alarms always occur when a fault occurs, resulting in a pattern that is very indicative of the underlying fault. This core set of alarms is repeated for every occurrence of a particular fault under different operating conditions.
- A few alarms occur repeatedly. These alarms represent innocuous events that occur during normal operation, and will probably not help in root cause analysis.

In this paper we present a method that exploits these properties to automatically associate patterns of change point based alarms with a given fault. Unlike earlier approaches [1], we can learn the problem signature for a fault with a very small number of fault instances. Our method also adaptively updates problem signatures as new information becomes available. Additionally, our method does not assume any prior domain-expert knowledge, and it learns effective problem signatures based only on feedback from the system administrator. Further, the problem signatures learned by our method are flexible and do not require exact matches to locate a root cause.

The layout of this paper is as follows. Section 2 presents related work. Section 3 describes our learning method. Section 4 describes our system design. Section 5 presents experimental results. Section 6 discusses future work and conclusions.

2 Related Work

The most common approaches to fault localization include AI techniques [3] such as rule-based techniques, model-based techniques, neural networks, decision trees, model traversing techniques such as dependency graphs [5][11] and fault propagation techniques [9] such as Bayesian networks and causality graphs.

As discussed in Section 1, automatic learning of rules has been studied earlier by Hellerstein et al. [1]. They discover patterns using association rule mining based techniques [14]. Additionally, each fault is usually associated with a specific pattern of events. Association rule based techniques require a large number of sample instances before discovering *"k-itemset"* [16] in a large number of events. The method presented in this paper overcomes this limitation and is able to discover patterns with very few fault instances. Another drawback of their technique is their reliance on pattern periodicity. Our method does not make any such assumption.

In another closely related work [9], the authors describe an event driven fault diagnosis technique that employs incremental learning. The authors propose techniques to rank a fault according to a "goodness" measure that allows multiple simultaneous faults to be identified. Fault diagnosis is incrementally improved as more symptoms become available. Although this technique is promising, it makes an assumption about the presence of a symptom-fault map as an input. Such a map may not be available in an enterprise environment. Our method makes no such assumption.

Several earlier approaches have used dependency analysis for fault localization. In [5][11] the authors assume that the mechanism to generate events is already in place and the root cause analysis algorithm analyzes these events in a systematic way using certain properties of the executing environment such as a dependency tree. Alarms relying on static dependencies between system components may be analyzed for problem determination [7]. Katker et al. [12] also shows how the dependency graph may be used to perform systematic analysis of a problem and identify the root cause in the network fault management domain. In both these approaches, the authors assume the presence of a dependency tree. These approaches may not work in dynamic enterprise systems where dependencies are ephemeral.

Other related work [10][6] has focused on studying the behavior of the various components and structural changes in the system and looking for anomalies in them. These approaches usually isolate the problem to one system component. Thus, they fall short of localizing the actual root cause and can only detect bottlenecks in the path of transactions. In [10], the incoming requests are traced and the list of the components used by several succeeded or failed requests are clustered to statistically identify the set of failed components. In [6], an optimized set of synthetic transactions is used to probe the system for possible problems. This technique puts additional load on the system which may not be acceptable to customers in a production environment. Further, constructing an optimized set of probes is an N-P hard problem.

In [8], a combination of probing (using fault injection) and dependency analysis is used for fault localization. Dependency information is generated by Active Dependency Discovery (ADD). ADD builds the system dependency graph by individually perturbing the system components during a testing phase, while fault injection is used at run-time. This technique suffers from similar disadvantages as [6].

Rule based systems such as [4] are used to define rules, and events are generated based on satisfaction of these rules. In classical rule based systems, rules are specified manually and they are static in nature i.e. they do not evolve automatically.

3 Learning Methodology

In this section we describe our method for learning patterns (or *problem signatures*[1]) corresponding to faults that occur in enterprise environments. We assume that no two

[1] We use the terms patterns, signatures and problem signatures interchangeably in this paper.

faults occur simultaneously. The learning method operates on the premise that when a fault occurs in a system, it is usually associated with a specific pattern of events. In our system, these events correspond to abrupt changes in performance metrics.

The input to our learning method comprises of:

a. A sequence of time-stamped events representing change point based alarms that arise from each application server in a clustered system;
b. Times of occurrence of faults at a given application server;
c. Input from a system administrator who correctly labels a fault when it occurs for the first time, or when the method fails to detect it altogether;
d. Feedback from a system administrator to verify the correctness of our output.

The mechanism to provide the first two inputs is described in Section 4. We first define two scores computed by our learner - *co-occurrence* score and *relevance* score, and then describe our learning and matching algorithm.

3.1 Co-occurrence Score

Our learning method computes a co-occurrence score, or *c-score,* for every alarm that is ever raised within a fixed time window around the occurrence of a fault. For a fault *F*, the *c-score* measures the probability of an alarm *A* being triggered when *F* occurs. The *c-score* is computed as follows

$$c = \frac{\#(A \,\&\, F)}{\# F}$$

Here $\#(A \,\&\, F)$ is the number of times *A* is raised when *F* occurs and $\# F$ is the total number of occurrences of *F*. The *c-score* for an alarm-fault pair ranges from 0 to 1. A high *c-score* indicates a high probability of *A* occurring when *F* occurs.

3.2 Relevance Score

Our learning method computes a relevance score, or *r-score,* for every single alarm that it ever encounters. The *r-score* for an alarm is a measure of the importance of the alarm as a fault indicator. An alarm has high relevance if it usually occurs only when a fault occurs. The *r-score* for an alarm A is computed as follows

$$r = \frac{\#(A \,\&\, Fault)}{\# A}$$

where $\#(A \,\&\, Fault)$ is the number of times *A* is raised when *any* fault occurs in the system, and $\# A$ is the total number of times *A* has been raised so far. The *r-score* for an alarm ranges from 0 to 1. Note that the *r-score* is a global value for an alarm i.e. there is just one *r-score* for an alarm unlike the *c-score* which is per alarm-fault pair.

An assumption here is that the system runs in normal mode more often than it does in faulty mode. When this is true, alarms raised regularly during normal operation have low *r-scores*, while alarms raised only when faults occur have high *r-scores*.

3.3 Learning and Matching Algorithm

We present here our method for learning and matching fault patterns. The method uses a *pattern repository* to store patterns that it learns. It starts with an empty repository and adds patterns based on administrator feedback. If a fault occurs when the repository is empty, our method just notifies the administrator that a fault has occurred. After locating the root cause, the administrator provides a new fault label[2]. Our method then records the alarm pattern observed around the fault, along with the fault label, as a new signature. Each alarm in this signature is assigned a *c-score* of 1.

For every subsequent fault occurrence, our method uses the following procedure in order to attempt a match with fault patterns that exist in the repository. Assume that S_F is the set of all the faults currently recorded in the repository. For each fault $F \in S_F$, let S_{AF} represent the set of all the alarms A that form the problem signature for F. Let each alarm $A \in S_{AF}$ have a *c-score* $C_{A|F}$, when associated with a fault F. Also, assume that the set of alarms associated with the currently observed fault in the system is S_C. For each fault $F \in S_F$, the learner computes two values, a *degree of match* and a *mismatch penalty*. The *degree of match* rewards F for every alarm in S_C that also occurs in S_{AF}. The *mismatch penalty* penalizes F for every alarm in S_C that does not occur in S_{AF}.

To compute the degree of match for a fault $F \in S_F$, the learning method first obtains an intersection set S_{CF} - a set of alarms common to S_{AF} and S_C

$$S_{CF} = S_{AF} \cap S_C .$$

It then computes the degree of match D_F as follows

$$D_F = \frac{\sum C_{A|F} \forall A \in S_{CF}}{\sum C_{A|F} \forall A \in S_{AF}}$$

The numerator in the above formula is the sum of the *c-scores* of alarms in the intersection set S_{CF}, and the denominator is the sum of the *c-scores* of alarms in S_{AF}. The ratio is thus a measure of how well S_C matches with S_{AF}. When a majority of alarms (that have a high *c-score)* in S_{AF} occur in S_C, D_F is high.

To compute the *mismatch penalty* for a fault $F \in S_F$, the learning method first obtains a difference set S_{MF} - a set of alarms that are in S_C but not in S_{AF}

$$S_{MF} = S_C - S_{AF}$$

It then computes the mismatch penalty as follows

$$M_F = 1 - \frac{\sum R_A \forall A \in S_{MF}}{\sum R_A \forall A \in S_C}$$

[2] Fault labels have a one to one correspondence with problem signatures in the repository.

The numerator in the second term for the M_F formula is the sum of the *r-scores* of alarms in S_{MF}, and the denominator is the sum of the *r-scores* of alarms in S_C. By definition, the *r-score* is high for relevant alarms and low for irrelevant alarms. Hence, if there are mostly irrelevant alarms in S_{MF}, the ratio in the second term would be very low and M_F would have a high value.

Using D_F and M_F we compute a final ranking weight W_F for a fault *F* as,

$$W_F = D_F * M_F$$

Once our method computes ranking weights for all faults in the repository, it presents to the administrator a sorted list of faults with weights above a threshold. If no fault in the repository has a weight above the threshold, it reports that there is no match.

The administrator uses this list to locate the fault causing the current performance problem. If the actual fault is found on the list, the administrator accepts the fault. This feedback is used by the learning method to update the *c-scores* for all alarms in S_C for that particular fault. If list does not contain the actual fault, the administrator rejects the list and assigns a new label to the fault. The learner then creates a new entry in the pattern repository, containing the alarms in S_C, each with a *c-score* of 1.

3.4 Matching Algorithm Example

We present here an example that explains the functioning of our method. Assume that S_F is the set of faults currently in the fault repository and $S_F = \{F_1, F_2, F_3\}$. These faults have the following signatures stored as sets of alarm and *c-score* pairs.

$S_{AF1} = \{(A_1,1.0),(A_2,1.0),(A_3,0.35)\}$, $S_{AF2} = \{(A_2,0.75),(A_4,1.0),(A_5,0.75)\}$

$S_{AF3} = \{(A_5,0.6),(A_6,1.0),(A_7,0.9)\}$

Suppose we now observe a fault with a set of alarms $S_C = \{A_1, A_2, A_4, A_6\}$. Assume that *r-scores* of these alarms are $R_{A1} = 0.4$, $R_{A2} = 1.0$, $R_{A4} = 0.9$, $R_{A6} = 0.45$.

The intersection of the alarms in S_C with S_{AF1}, S_{AF2} and S_{AF3} yields the sets $S_{CF1} = \{A_1, A_2\}$, $S_{CF2} = \{A_2, A_4\}$ and $S_{CF3} = \{A_6\}$

The degree of match for each problem signature is computed as

$$D_{F1} = \frac{1.0 + 1.0}{1.0 + 1.0 + 0.35} = 0.85, \ D_{F2} = 0.7 \text{ and } D_{F3} = 0.4$$

For mismatch penalties, we compute the difference of set S_C from S_{AF1}, S_{AF2}, S_{AF3} to obtain $S_{MF1} = \{A_4, A_6\}$, $S_{MF2} = \{A_1, A_6\}$ and $S_{MF3} = \{A_1, A_2, A_4\}$.

The mismatch penalties are

$$M_{F1} = 1 - \frac{0.9 + 0.45}{0.4 + 1.0 + 0.9 + 0.45} = 0.51, \ M_{F2} = 0.69 \text{ and } M_{F3} = 0.16$$

The ranking weights are $W_{F1} = 0.85 * 0.51 = 0.43$, $W_{F2} = 0.48$, $W_{F3} = 0.06$. With a weight threshold of 0.4, the output list is F_2, F_1. Note that even though F_1 has a higher degree of match than F_2, F_1 is second on the list due to a higher mismatch penalty.

4 System Design

We describe here our system design for providing inputs to the learning method. The first input required by our method is a sequence of time-stamped alarms for each server in the cluster. For this, we monitor and sample runtime performance metrics at each server and use change point detection techniques such as difference of means [2] to generate alarms. A learning component is implemented on each server, and a pattern repository is shared amongst all learning components. The trigger for the method comes from an SLA breach predictor (*SBP)* operating at each server.

The *SBP* triggers the learning method when it detects an abrupt change in response time or throughput in the absence of any significant change in the input load on a server. Once the learning component gets a trigger from the *SBP*, it fetches all the alarms in a fixed time window around the current trigger. These alarms are then fed to the learning method and it operates on them as described in Section 3. The output from the learning method is a list of faults sorted in order of relevance. This list of faults is sent to a central controller which takes one of the following actions:

a. If only one server reports a list of faults during a given time interval, a single list is displayed to the administrator along with the name of the affected server.
b. If all running servers report a list of faults during a given time interval and the most relevant fault is the same for all servers, it is assumed that the fault is at a resource shared by all the servers, typically a database system. The controller chooses the most relevant fault and displays that fault to the administrator.
c. If a subset of running servers report a list of faults during a given time interval, this could either be caused by multiple independent faults or by a fault that occurred on one server and has affected the runtime metrics of other servers due to an “interference effect”. In our current design, the controller treats the two cases in the same manner and displays the lists for all affected servers.

5 Evaluation

We describe in this section, our test-bed, three-tier application and workload generator, system implementation, and our experimental results.

5.1 Test-Bed, Application and Workload

Our test-bed consists of eight machines: one machine hosting two load generators, two request router machines, three application server machines, a relational database server machine, and a machine that hosts the cluster management server. The back end servers form a cluster, and the workload arriving at the routers is distributed to these servers based on a dynamic routing weight assigned to each server. The machines running the back end servers have identical configurations. They have a single 2.66GHz Pentium4 CPU and 1GB RAM. The machine running the workload generators is identical except that it has 2GB RAM. Each of the routers have one 1.7GHz Intel Xeon CPU and 1GB RAM. The database machine has one 2.8GHz Intel Xeon CPU and 2GB RAM. All machines run RedHat Linux Enterprise Edition 3, kernel version 2.4.21-27.0.1.EL. The router and back end servers run the IBM WebSphere middleware platform, and the database server runs DB2 8.1.

For our experiments, we ran Trade 6 [17] on each of the servers. Trade 6 is an end-to-end benchmark that models a brokerage application. It provides an application mix of servlets, JSPs, enterprise beans, message-driven beans, JDBC and JMS data access. It supports operations provided by a typical stock brokerage application.

We used IBM WebSphere Workload Simulator [18] to drive our experiments. The workload consists of multiple clients concurrently performing a series of operations on their accounts over multiple sessions. Each of the clients has a think time of 1 second. The actions performed by each client and the corresponding probabilities of their invocation are: *register new user* (2%), *view account home page* (20%), *view account details* (10%), *update account* (4%), *view portfolio* (12%), *browse stock quotes* (40%), *stock buy* (4%), *stock sell* (4%), and *logoff* (4%). These values correspond to the typical usage pattern of a trading application.

5.2 Experimental Runs

In order to perform a detailed evaluation of our learning method over a number of parameters and fault instances, we generated traces containing the inputs required by our method and performed an offline analysis. The only difference from an online version is that the administrator feedback was provided as part of the experimentation.

We implemented the breach predictor as a component that resides within one of the routers in our test-bed. It subscribed to router statistics and logged response time information per server at a 5 second interval. Each server in the cluster also monitored and logged performance metric information. We ran a total of 60 experiments, each of duration one hour (45 minutes of normal operation followed by a fault). The five faults that we randomly inserted in our system were:

- CPU hogging process at a node hosting an application server
- Application server hang (created by causing requests to sleep)
- Application server to database network failure (simulated using Linux *iptables*)
- Database shutdown
- Database performance problem (created either by a CPU hog or an index drop).

We maintained a constant client load during individual experiments, and varied the load between 30 and 400 clients across experiments. After obtaining the traces for 60 experiments, the learning and matching phase involved feeding these traces to our method sequentially. This phase presents a specific sequence of alarms to the learning method. In order to avoid any bias towards a particular sequence of alarms, we repeated this phase a 100 times, providing a different random ordering of the traces each time. For all our experiments we used a *c-score* threshold of 0.5.

5.3 False Positives and Negatives

We first explore the performance of our learning method in terms of false positives and negatives. We compute the false negative count as the number of times our method does not recognize a fault. However, when our method sees a fault for the first time, it does not count as a false negative. After completing all 100 runs, we compute the average number of false negatives generated by our method.

False positives occur when a newly introduced fault is recognized as an existing fault. We use the following methodology to estimate false positives. We randomly choose a fault *F* and remove all traces containing *F* from the learning phase. We then feed traces containing *F* to our method and calculate the number of times it is recognized as an already observed fault. We repeat this procedure for each fault and compute the average number of false positives.

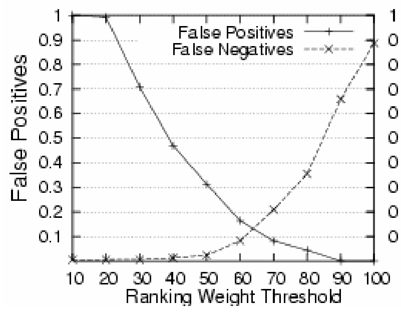


Fig. 1. False positives and negatives

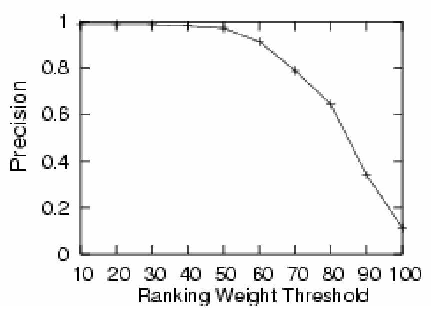


Fig. 2. Precision

Figure 1 shows the average percent of false positives and false negatives generated by our method as we vary the ranking weight threshold between 10 and 100. Recall that the ranking weight is our estimate of the confidence that a new fault pattern matches with a pattern in our repository. Only pattern matches resulting in a ranking weight above the threshold are displayed to the administrator.

As one would expect, when the threshold is low (20% or lower) we generate a large number of false positives. This is because at low thresholds even irrelevant faults are likely to generate a match. As we increase the threshold beyond 20%, the number of false positives drops steadily, and it is close to zero at high thresholds (80% or higher). Note that false positives are generated only when a new fault occurs in the system. Since new faults can be considered to have relatively low occurrence over a long run of a system, a false positive percent of 20-30% may also be acceptable after an initial learning period. Our method generates few false negatives for thresholds under 50%. For thresholds in the 50-70% range, false negatives range from 3-21%. Thresholds over 70% generate a high percent of false negatives.

Hence, there is a trade off between the number of false positives and negatives. The curves for the two measures intersect when the ranking weight threshold is about 65%, and the percent of false positives and negatives is each about 13%. A good region of operation for our method is between a weight threshold of 50-65%, with more false positives at the lower end, and more false negatives at the higher end. An approach that we can use to obtain good overall performance is to start our method using a threshold close to 65%. During this initial phase, it is likely that a fault occurring in the system will be new, and the high threshold will help in generating few false positives. As our method learns patterns, and new faults become relatively rare, the threshold can be lowered to 50% in order to reduce false negatives.

5.4 Precision

If a fault is always detected but usually ends up at the bottom of the list of potential root causes, the analysis is likely to be of little use. In order to measure how effectively our method matches new instances of known faults, we define a *precision* measure. Each time our method detects a fault, we compute a precision score using the formula $\frac{(\#F - i - 1)}{\#F}$, where *#F* is the number of faults in the repository, and *i* is the position of the actual fault in the output list. A false negative is assigned a precision of 0, and our method is not penalized for new faults not present in the repository. We perform 100 iterations over the traces using the random orderings described above, and compute the average precision.

Figure 2 shows average precision values for ranking weight thresholds ranging from 10-100. We can see that our precision score is high for thresholds ranging from 10-60%. For thresholds ranging from 10-30%, the average precision is 98.7%. At a threshold of 50% the precision is 97%, and at a threshold of 70% the precision is 79%. These numbers correspond well with the false negative numbers presented in the previous section, and they indicate that when the method detects a fault, it usually places the correct fault at the top of the list of potential faults.

5.5 Rate of Learning

We demonstrate in this section a key property of our learning method – it can learn a relevant pattern for a fault given very few instances of the fault. To evaluate the rate at which our method learns patterns, we perform the following experiments. We first set a *learning threshold* which is the maximum number of instances of a fault that our method can use in order to learn. Any fault instances over the learning threshold are only used to evaluate the precision of our method, and cannot be used by the learner to update its scores. We then run our method over each fault using several values of the learning threshold, and obtain an average precision score for each threshold value.

Figure 3 shows precision scores for three values of the learning threshold, 1, 2, and 4. The precision values are shown for ranking weight thresholds ranging from 10-100. We can see that when our method is provided with only a single instance of a fault, it has precision values of about 90% when the ranking weight is 50%. This is only about 8% worse than the best possible precision score. At a ranking weight threshold of 70%, the precision is about 14% lower than the best possible precision.

This data clearly shows that our method learns patterns rapidly, with as few as two instances of each fault required to obtain high precision. This is largely due to two reasons. First, we use change point detection techniques to generate events and we have found that they reliably generate unique patterns for different faults. Second, the *c-score* and the *r-score* used by our method help us filter out spurious events.

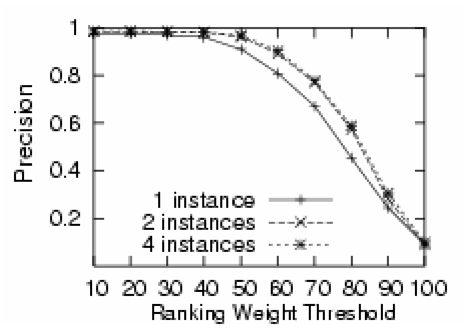


Fig. 3. Rate of Learning

6 Conclusions and Future Work

In this paper, we presented a novel technique for discovering change point based adaptive patterns for problem resolution in enterprise systems. We demonstrated the efficacy of our technique by learning the problem signatures for five common faults that occur in enterprise systems and reliably recognizing these problems with high precision. One of the main contributions of this paper is that we discover these patterns quickly, with few fault instances. This is a significant improvement over traditional data mining techniques which require a large number of fault instances to discover patterns. The patterns generated by our method are flexible, in that they do not require exact matches for triggering. Another significant contribution of our work is that our technique can discover adaptive patterns i.e. if a fault pattern changes over time due to reasons such as changes in topology, workload, application version, our method automatically updates the pattern repository with the new pattern over time.

There are a few future directions to the work presented in this paper. One of the issues that we intend to tackle is the absence of certain alarms during the problematic phase. The absence of a particular alarm during the problematic phase may be as indicative of a fault as the presence of other alarms. Our method currently does not handle cases where significantly different patterns are generated for a single fault. An extension to our method would associate more than one pattern to a fault if there is a significant mismatch between the patterns. Another improvement to our technique is the use of negative feedback from the administrator. In our future research, we also intend to include events generated by sources other than the currently monitored performance metrics, such as event logs.

References

1. Hellerstein J. L., Ma S., Perng C.: Discovering Actionable Patterns in Event Data. IBM Systems Journal, Vol 41, No 3, 2002.
2. Agarwal M., Gupta M., Mann V., Sachindran N., Anerousis N., Mummert L.: Problem Determination in Enterprise Middleware Systems using Change Point Correlation of Time Series Data. 9thIEEE/IFIP Network Operations and Management Symposium (NOMS), Vancouver, Canada, May 2006.

3. Steinder M., Sethi A.:The present and future of event correlation: A need for end-to-end service fault localization. SCI-2001, 5th World Multiconference on Systemics, Cybernetics, and Informatics, Orlando, FL (July 2001), pp. 124-129
4. Appleby K., Goldszmidt G., Steinder M.: Yemanja A Layered Fault Localization System for Multi-domain Computing Utilities. IM 2001
5. Gruschke B.: Integrated Event Management: Event Correlation Using Dependency Graphs. DSOM 1998.
6. Brodie M., Rish I., Ma S., Odintsova N.: Active Probing Strategies for Problem Diagnosis in Distributed Systems. IJCAI 2003
7. Gao J., Kar G., Kermani P.: Approaches to Building Self Healing Systems using Dependency Analysis. IEEE/IFIP Network Operations and Management Symposium (NOMS), April, 2004.
8. Brown A., Kar G., Keller A.: An Active Approach to Characterizing Dynamic Dependencies for Problem Determination in a Distributed Environment. IM 2001.
9. Steinder M., Sethi A.: Non-deterministic Event-driven Fault Diagnosis through Incremental Hypothesis Updating, In Integrated Network Management, VIII} (G. Goldszmidt and J. Schonwalder (eds.)), pp. 635-648, Boston, MA: Kluwer Academic Publishers, 2003
10. M. Y. Chen, E. Kıcıman, E. Fratkin, A. Fox, E. Brewer: Pinpoint: PD in Large, Dynamic Internet Services, International Conference on Dependable Systems and Networks (DSN'02), 2002.
11. Choi J., Choi M., Lee S.: An Alarm Correlation and Fault Identification Scheme Based on OSI Managed Object Classes. IEEE International Conference on Communications, Vancouver, BC, Canada, 1999, pp. 1547–51.
12. Katker S., Paterok M.: Fault Isolation and Event Correlation for Integrated Fault Management. Integrated Network Management V, Chapman and Hall, May 1997.
13. Aguilera M. et.al.: Performance Debugging for Distributed Systems of Black Boxes. 19th ACM Symposium on Operating Systems Principles, October 2003.
14. Agarwal R., Imielinski T., and Swami A.: Mining association rules between sets of items in large databases. ACM SIGMOD Conference on Management of Data, pp. 207-216, May 1993.
15. Agarwal M., Appleby K., Faik J., Kar G., Neogi A., Sailer A.: Threshold management for Problem Determination in Transaction Oriented e-Commerce Systems., 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005), May 2005.
16. Fu A., Kwong R., Tang J., "Mining N most interesting Itemsets" 12th International Symposium on Methodologies for Intelligent Systems (ISMIS), Springer-Verlag, LNCS, Charlotte, North Carolina, USA, Oct 11-14, 2000
17. IBM Trade Performance Benchmark Sample, http://www-306.ibm.com/software/ webservers/ appserv/was/performance.html
18. IBM Websphere Studio Workload Simulator, http://www-306.ibm.com/software/awdtools/ studioworkloadsimulator/

Business-Driven Decision Support for Change Management: Planning and Scheduling of Changes

Jacques Sauvé[1], Rodrigo Rebouças[1], Antão Moura[1], Claudio Bartolini[2], Abdel Boulmakoul[3], and David Trastour[3]

[1] Departamento de Sistemas e Computação - University of Campina Grande (UFCG), Brazil
[2] HP Laboratories Palo Alto, USA
[3] HP Laboratories Bristol, UK
{jacques, rodrigor, antao}@dsc.ufcg.edu.br,
{claudio.bartolini, abdel.boulmakoul, david.trastour}@hp.com

Abstract. From the results of a web survey we carried out in 2006, the main challenge in IT change management from a change manager's perspective was identified as *planning and scheduling of changes*. This paper begins to address this problem by taking business considerations into account; this is done through a business-driven IT management (BDIM) approach. A reference architecture that follows BDIM principles is sketched; it includes a mathematical model linking IT availability metrics to business objectives. Monetary loss due to service level violations on service availability is used as the main business metric. We present a numerical illustration of how the derived metrics may support change management decisions in order to plan and schedule changes to minimize adverse business impact.

Keywords: Change management, business-driven IT management, service level management, Information Technology Infrastructure Library (ITIL), business metrics, modeling, performance evaluation, business impact, decision and negotiation support tools.

1 Introduction

IT management has become more user-centric and less service provider-dependent with the popularity of the practices recommended by the Information Technology Infrastructure Library – ITIL [3], which is used as the basis for the IT Service Management framework – ITSM [8]. ITSM defines a number of processes that are organized into 5 modules: security management; IT & communication infrastructure management; application management; service support (incident, problem, configuration, change and release management processes); and service delivery (service level, capacity, availability, continuity and financial management for IT services). Within the realm of ITSM, this paper focuses on the *change management* process.

ITSM expresses goals and gives guidelines to IT managers for ensuring smooth running of IT service delivery and support. For instance, the mission of the change management process is defined as “[ensuring] that **standardized methods and procedures** are used for efficient and **prompt handling of all changes**, in order to **minimize the impact** of any related incidents upon service” [3]. However, it falls short of defining control objectives for IT. This shortcoming is addressed by the COBIT framework (Control Objectives for Information and related Technologies) [2]. In order to gauge the maturity and quality of IT service delivery and support activities, COBIT introduces a number of key performance indicators (KPIs) that drive the process goals, which in turn are measured by process key goal indicators (KGIs). Examples of key performance indicators for the change management process are the number of emergency changes, or the number of changes that were rolled back, in a change management context. For activities in the service delivery scope, such as service level management, metrics such as service availability and reliability are taken into account.

The first wave of management software (from the early 90’s), concentrated on monitoring availability, resource consumptions levels, etc. In the last three or four years, software tools have appeared that help with other IT management activities, in particular with help desk and IT service support. These tools provide valuable help to IT managers in making informed decisions on the actions to take to ensure the smooth running of IT processes.

However, just because the IT systems are running smoothly, it does not follow that the business that IT supports is best served by it. In order to ensure business-IT alignment, metrics should be taken into account which are more business-oriented in nature, such as cost, revenue or financial loss. This consideration is the basis for the Business–Driven IT Management (BDIM) discipline [6]. BDIM steers ITSM towards business alignment, i.e., to contribute to business results. This paper uses a BDIM approach to address change management challenges.

BDIM attempts to gauge the impact that IT has on the business and aims at rethinking IT management from this perspective. BDIM involves a new culture, tools and decision–making processes that aim to *help the business*. A complete ITSM shift to BDIM requires IT personnel or automated tools to use business metrics to gauge the QoS offered to a business user. Although BDIM has been attracting mounting research efforts, attempts at investigating the feasibility and options of spreading BDIM applications to cover ITIL management processes are still scarce. Some recent applications include incident prioritization [1], capacity planning [5], and automatic change management process [4]. Embedding results of such efforts in tools for automating decision and negotiation support is at its very beginning. This is particularly true for the case of human-assisted change management processes. This paper proposes a BDIM-based solution which could be embedded in a tool to support decision and negotiation activities in a more generic, ITSM–based change management process.

The remainder of the paper is organized into sections 2 through 7. Section 2 discusses current change management challenges as elicited by a survey performed in early 2006. Section 3 begins to address some of these challenges by describing a layered reference architecture for business-driven IT change management (BDIM-CM) solutions. Section 4 details how metrics for the BDIM-CM solution may be derived. Section 5 presents a numerical illustration on how the derived metrics may support change management decisions in order to minimize adverse business impact. Section 6 briefly examines competitive and related work, including the few tools available on the market. In section 7 we draw our conclusions and give a preview of our further work in this space.

2 Major Challenges in Change Management

The change management process comprises four groups of activities:

- Request For Change (RFC) acceptance, classification and processing;
- approval, planning of changes;
- execution, tests and reversal of changes;
- change evaluation.

Current state-of-the-practice solutions for change management suffer from several acute problems, including the volume of changes, change complexity and inappropriate tools. It appears that the most demanding challenges faced by technical personnel in charge of change management lie in activities from the first two groups. ITIL recommends that change classification be done according to change priority and change category (which components are affected). *Priority* is set according to the business importance of an RFC relative to other RFCs; *category* is determined based on the availability of resources, risk to services and on the impact of the changes. *Planning* items include scheduling, allocation of resources, budgeting, sequencing of activities, back out plans and communication. According to results from a Web questionnaire posted in the first quarter of 2006 [7], particular attention should be devoted to planning and scheduling issues. The questionnaire respondents were ITSM practitioners, all engaged in change management (some with over 10 years of experience), from 21 companies worldwide. Seventeen of these companies already have change management processes in place (11 use ITIL and 6 adopt other practices); four are just starting to implement change management. Nine of the companies are in the business of providing IT services (including consulting), 4 are telecoms, 4 are in financial services and the others are either in government, health care or manufacturing. Seven companies have yearly revenues over US$ 1 billion while eight make under US$ 10 million annually. Questionnaire respondents who follow ITIL change management process recommendations ranked the first 3 most important change management challenges as being (Figure 1):

1. scheduling/planning changes (with 47 points out of a maximum 55, or over 85%);
2. high number of emergency changes (43 points or 78%); and
3. RFC scope ill-definition (40 points or 72%).

The survey also indicates that:

- adopters of an ad-hoc change management process (as opposed to a formal process like ITIL's) rank planning/scheduling as the most important challenge (80%), together with "unauthorized changes"; unauthorized changes are a problem to be expected in an ad-hoc process

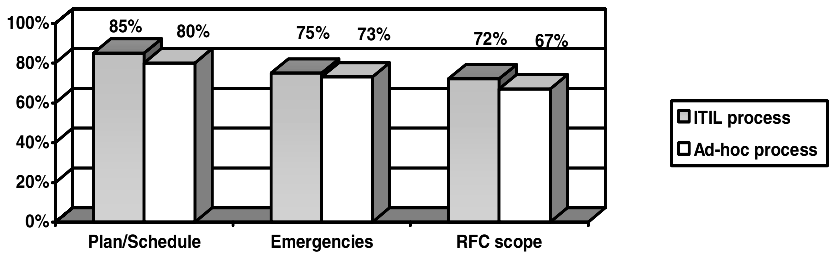


Fig. 1. Most frequent problems in Change Management

- "high number of emergency changes" is ranked second (73%) together with "notification of people affected";
- "RFC scope mal-definition" is ranked third (67%), but together with "inconsistent Configuration Management DataBase (CMDB)" (again, a possibly symptom of an ad-hoc process).

This paper contributes to addressing the most critical issues of change planning/scheduling to minimize negative impact to a service provider's business.

Properly addressing change planning and scheduling challenges is no trivial endeavor. As commented by one of the survey respondents, "scheduling is non-trivial due to people and process problems". Changes take place in a very dynamic environment: people become unavailable, business conditions vary and "urgent changes" may materialize. Hence, change plans and schedules have to be adjusted correspondingly. A change manager may have to build and consider several plans/schedules before a given plan is actually implemented. In an outsourcing environment, negotiating change windows with business clients is another complicating, human-dependent factor. Typically, outsourcing agreements do not provide explicit information on feasible time windows for scheduling changes that affect the associated service. Since no contractual binding exists, windows may be (and are) re-negotiated, causing re-planning and re-scheduling. The sheer volume of requests for change – RFCs – makes the scheduling exercise very complicated. As an example, the HP Managed Services organization handles 300 to 400 RFCs per weekend for a single customer. Therefore change classification and planning are currently driven by technical issues with little consideration for business needs or priorities. The solution for (re-) scheduling and elaborating such diverse plans that we begin to sketch in the next section can ease the lives of those responsible for the change management process.

3 Business-Driven Planning and Scheduling of Changes

Figure 2 depicts a reference architecture for BDIM solutions for change management, built upon a three-layer hierarchical model.

The bottom part of the figure shows objects from the IT and business environments; on the IT side, this can include, for example, RFCs and the Configuration Management Data Base (CMDB). Since a Service Level Agreement

(SLA) captures the business requirements imposed on the IT function, it sits at the boundary between the two environments. From its inputs, the lower layer uses *business-IT linkage models* to produce *IT-business linkage metrics* in the sense defined in [6]: metrics that numerically capture relationships between IT causes and their effects on business results. Examples of such metrics are: risk of adversely affecting business operations if a change is not successful and the impact of an unsuccessful or delayed change (such as potential financial loss). As will be seen in the next section's examples, these metrics are calculated from the probability of violating an SLA, which in turn may depend on the change schedule. The solution discussed here adopts *impact* as the linkage metric and uses the probability of SLA violation to estimate it.

Linkage metrics are then fed to the middle layer – labeled *decision support* - where decisions are made and used to steer activities of the change management process (to the left of Figure 2) and/or to help negotiate change management process details – such as change windows – with the IT client (at the top of the figure – *negotiation support*).

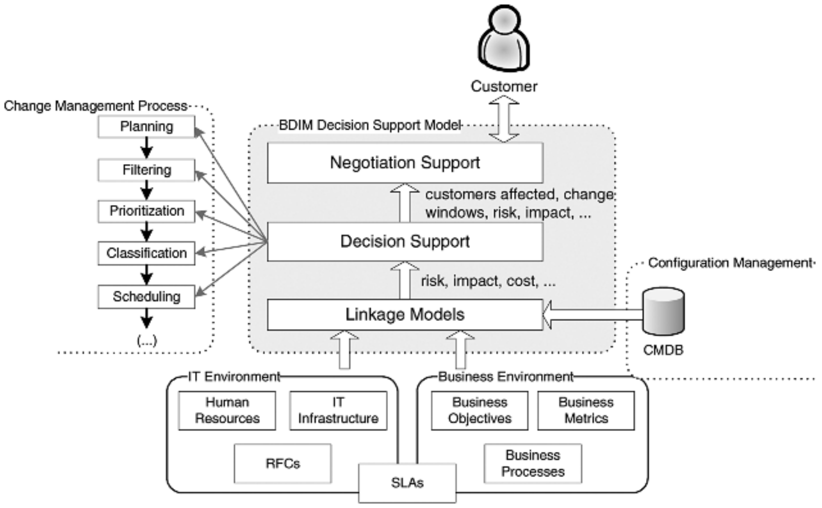


Fig. 2. A hierarchical model for business-driven change management solutions

We propose to use the above reference architecture in a change management setting according to the following BDIM approach. By knowing details of a given RFC such as the affected components (called *configuration items* or CIs by ITIL), the past history of service levels and service level objectives (SLOs), one can calculate the probability of violating a given SLA if the change is implemented at a given time. The business-IT linkage metrics produced by the linkage layer allow one to determine the expected monetary loss that will potentially result from a given change schedule. As a result, the change manager now has numerical business impact estimates from which to choose the changes that should be implemented during a particular change window. The past comments assume a known change window time. When this time must be negotiated with the client, impact metrics will be helpful in choosing appropriate time windows to perform changes. The business perspective introduced through the business-IT linkage model eases negotiations because arguments are presented to the client in familiar business terms (in this case, as potential financial loss).

4 Business-IT Linkage Model

This section develops a model to capture the impact of changes on the business. Consider a scenario where a service organization provides support for IT services subject to SLAs including an SLO that states a minimum availability. The client organizations use the services to process “revenue-generating sessions”; an example of such a service could be an e-commerce site where site visitors generate buying transactions during sessions accessing the service.. The provider earns a fixed fee for each successfully completed session and pays a penalty whenever the SLA is violated. Whenever a service is down but the associated SLA is not (yet) violated, the provider stops collecting fees on that service since no session can be serviced. If the SLA is violated however, besides losing the contracted fees, the provider must pay a penalty to the client at the end of the SLA evaluation period.

The business objective that we consider is to minimize the financial business loss incurred by the provider due to imperfections of supporting IT services. Changes to the IT infrastructure are “imperfections” in the sense that they may force a service to be brought down to perform the changes; as a result, changes can cause business loss. In our example, the episodes that can have an impact on the business loss are *violations of the minimum availability SLO* and *system downtime* due to changes.

- **Violation of SLO on minimum availability.** Due to penalties included as SLA clauses and, more importantly, in order not to tarnish the service provider’s image, SLA violation is frequently cited by change managers as a prime driver for decision making during change planning and scheduling.
- **System downtime.** In our example, whenever the service is down and the availability SLO is not violated, the provider stops collecting fees since no session can be serviced. This has a direct impact on the business loss.

Next, we will estimate the likelihood and extent of impacting episodes due to requested changes (subsection 4.1), and derive their impact on the business loss (4.2).

4.1 Probability of SLA Violation and Extent of System Downtime

In order to calculate business loss – in the next section – expressions for the probability of SLA violation and the extent of system downtime must be obtained.

Before formalizing the analysis, let us informally explain what we seek. Imagine that the change manager has a set of changes that may be implemented in the current SLA evaluation period and that the SLA contains an SLO on service availability. Of this evaluation period (which has duration T), duration *t* has already elapsed and the change manager knows how service availability is standing up so far; in other words, the past is known. The future is *not* known but may be estimated: certain changes may be performed (or not) and they may bring down service, thus affecting the availability metric. Given the knowledge of all that has occurred in the past, the set of changes that may be considered and estimates of future availability, which changes should the change manager choose to perform in the current SLA evaluation period so as to minimize business loss?

We now formalize the problem and provide a solution. Let us first consider a single IT service s_j from the provider's set of services, $S=\{s_1,...,s_{|S|}\}$ and assume that the associated Service Level Agreement (SLA) in force for s_j has a Service Level Objective (SLO) on *availability*, A_j^{min}. Let the *mean service availability* for s_j be calculated over an evaluation period T [1], as determined by the associated SLA. This mean availability takes on a different value over each evaluation period and it is thus a random variable, denoted by $\tilde{A}_j$. We indicate the cumulative distribution of the service availability random variable with $F_j(x) = \Pr[\tilde{A}_j \leq x]$. Without loss of generality, let the current evaluation for availability start at time 0 and end at time T. Let us examine the situation at a point in time, t, s.t. $0 \leq t \leq T$ when the change manager must make scheduling decisions. Let the availability over period (t_1,t_2) be $A_j(t_1, t_2)$. The past mean availability over the time period $[0,t]$ is known (it is measured) and is simply: $A_j(0,t)$. The future mean availability over time period $[t,T]$ is $A_j(t, T)$. Finally the overall availability over the whole SLA evaluation period is $\tilde{A}_j = A_j(0, T)$. Now, we ask: "At time t, what is the probability that the availability threshold, A_j^{min}, specified in the SLA will be violated by time T?" The mean availability, $A_j(0, T)$, over the whole evaluation period, $[0, T]$, can be calculated from the mean values of past and future availability by summing up the uptime over both time periods:

$$\tilde{A}_j = A_j(0,T) = \frac{uptime}{T} = \frac{A_j(0,t)t + A_j(t,T)(T-t)}{T} \tag{1}$$

In the above, the term $A_j(0,t)t$ is the uptime accumulated in the past and $A_j(t,T)(T-t)$ is the expected future uptime. Then the distribution for availability, given that time has reached t, follows:

$$\Pr\left[A_j(0,T) \leq x \middle| t\right] = \Pr\left[A_j(t,T) \leq \frac{xT - A_j(0,t)t}{T-t}\right] \tag{2}$$

Given that the probability distribution for availability is assumed to be the same over any time period in the interval $[0,T]$, we can now express $V_j(t, T, A_j^{min})$, the probability, at time t, of violating the availability SLO for service s_j by time T:

$$V_j(t,T,A_j^{\min}) = \Pr\left[A_j(0,T) \leq A_j^{\min} \middle| t\right] = F_j\left(\frac{A_j^{\min}T - A_j(0,t)t}{T-t}\right) \tag{3}$$

[1]For simplicity we assume that all services have this same evaluation period.

This result does not take into account the fact that, in the future, changes will be implemented, that these changes may bring down the service and that the situation for availability is actually worse that that given above. We now turn our attention to the impact of changes affecting service s_j. Let $C = \{c_1, \ldots, c_{|C|}\}$ be the set of all changes to be considered by the change manager and let $T_n^c \subseteq [0,T]$ be the time interval during which a given change c_n in C is performed. Notice that implementing c_n may or may not affect s_j availability. Let service s_j be provisioned with a set, I_j^s, of components (Configuration Items or CIs, in ITIL parlance). If we let $I=\{i_1,\ldots,i_{|I|}\}$ be the set of all CIs in the CMDB, then $I_j^s \subseteq I$. Each change is subject to a plan that specifies the time at which the change implementation will start, and a subset $I_n^c \subseteq I$ of the CIs that will be affected by change c_n. The plan specifies which CIs will be brought down and when, so that one can calculate the time at which service will be brought down (if it is not already down) and the time at which it will be available again. The set of all intervals during which service s_j becomes unavailable within T is given by the union of all s_j–affecting change intervals, T_n^c, i.e.,

$$T_j^s = \bigcup_{\forall n | I_j^s \cap I_n^c \neq \emptyset} T_n^c \tag{4}$$

Now let ΔT_j^s (a scalar) correspond to the total time period during which s_j is unavailable, i.e., the sum (disconsidering overlaps) of the durations of all change intervals in T_j^s. Notice that all changes affecting service s_j will be implemented *after* the present moment (time t) so that any service downtime will need to be added to the "future" part of the evaluation period. Observing that the time period between $[t,T]$ but outside the service downtime called for in the change plan still obeys the same distribution of availability, we conclude that, with the changes planned for the current evaluation period, the probability of violating the SLA for service s_j is:

$$V_j(t,T,A_j^{\min}) = F_j\left(\frac{A_j^{\min}T - A_j(0,t)t}{T - t - \Delta T_j^s}\right) \tag{5}$$

In order to conclude the development, we need the cumulative probability distribution function, $F_j(x) = \Pr[\tilde{A}_j \le x]$. A result from reliability theory [9] states that, when the uptime (time-to-failure) and downtime (repair times) are exponentially distributed, availability follows the two-parameter Beta distribution with parameters α and β. The mean value for availability is simply $E[\tilde{A}_j] = \alpha/(\alpha+\beta)$. α and β are chosen to match historical availability distribution data. Typical values are $\alpha = 7$ and $\beta = 0.03$, yielding 99.57% availability averaged over several evaluation periods.

4.2 Impact of SLA Violations on Business Loss

We can now turn our attention to estimating business loss due to a single service s_j. Table 1 summarizes the parameters of the provider's revenue model.

Table 1. Provider's parameters for service s_j

Principal impact function variables	
$V_j(t,T,A_j^{\min})$	Probability of SLA violation, given the knowledge available at time *t*
ΔT_j^s	Service s_j unavailability period due to implementation of change selected by the change manager
Other impact function parameters	
π_j	Penalty ($) for service s_j SLA violation
T	SLA evaluation period for s_j
γ_j	Session throughput for service s_j
σ_j	Fixed fee ($) per successful session, for service s_j

At time *t*, the expected provider financial loss for service s_j during the current SLA evaluation period ending at time *T* is given by:

$$L_j^p(t,T) = V_j(t,T,A_j^{\min})\cdot\pi_j + \gamma_j\cdot\Delta T_j^s\cdot\sigma_j \tag{6}$$

Notice that loss is a function of the time at which decisions are made (*t*) since, as time passes, the duration of the "past" (period [0,*t*]) becomes larger, and availability becomes more and more defined by what happened in the past.

In the case of multiple services $S=\{s_1,.s_j..,s_{|S|}\}$ supported by the provider and affected by multiple changes, the total expected provider loss at time *t* is:

$$Loss_s(t,T) = \sum_{j=1}^{|s|} L_j^p(t,T) \tag{7}$$

5 Numerical Illustration of BDIM Support in Change Management

Consider a scenario where a service provider offers services $S=\{s_1,s_2,s_3\}$: s_1 is a web auction service; s_2 an e-commerce service; and s_3 a database service. At time t = day 10, three changes appear on the change manager's desk: service s_1 is to be brought down due to two changes to two of its supporting CIs: an operating system (CI_1), whose change (c_1) is to be implemented in 3 hours; and, a DataBase Management System (CI_2), whose version upgrade (change c_2) is expected to last 2 hours. Furthermore, services s_2 and s_3 share a firewall (CI_3) and do not use CI_1 and CI_2. A firewall change (c_3) is expected to last 4 hours. Table 2 lists SLA parameters for these three services as well as the availability situation at day 10. Let T = 30 days (monthly SLA evaluation).

Due to staff limitations, the change manager cannot do both $\{c_1,c_2\}$ and $\{c_3\}$ simultaneously. He must choose which set of changes to implement during the current SLA evaluation period: should changes $\{c_1,c_2\}$ or $\{c_3\}$ be done? Assume that there is no overlap possible between c_1 and c_2. Table 2 indicates that s_1 is the service with the greatest revenue stream ($24/s) and has good (past) availability. The combined duration of the changes for this service (3 + 2 = 5 hours) will not cause SLA violation over the current SLA evaluation period (at *t*=30 days). On the other hand, choosing to implement c_3 will cause the service s_2 SLA to be violated, making the provider pay a $10,000 penalty; SLA for service s_3 will not be violated. If one disregards all other change schedule influencing factors – such as political pressure from clients, concerns with provider image, change roll-back problems and the cost of not executing a given change – and if one simply analyzes SLA clauses and short-term past history, one may be tempted to opt for implementing $\{c_1,c_2\}$, affecting s_1 since this option is likely to yield a smaller loss. This is an approach commonly employed by change managers (avoid SLA violations!).

Table 2. Service configuration

Input	Service s_1	Service s_2	Service s_3
γ_j (session/s)	16	12	13
σ_j	$ 1.5	$ 0.7	$ 0.8
π_j	$ 30,000	$ 15,000	$ 10,000
$A_j^{\min}$	0.99	0.99	0.99
$A_j(0, day\ 10)$	0.998	0.992	0.9998

Let us now examine the situation in the light of business loss metrics (Figure 3). This figure compares business loss for both alternatives (choosing $\{c_1,c_2\}$or $\{ c_3\}$) as the decision time point changes (horizontal axis). This figure shows that, at day 10, selecting change $\{c_3\}$ is preferable since it causes lower business loss (around $14,000 compared to $21,000). However, should the three changes land on the change manager's desk for decision on day 17, say, it would be more advisable to select $\{c_1,c_2\}$ since expected loss for change $\{c_3\}$ has now increased substantially (to $27,000). The reason for this drastic change around day 16 is that, from that time on, change $\{c_3\}$ will cause an SLA violation for service s_3. Change management decisions are dynamic in nature and our business-IT linkage metrics captures this dynamic behavior.

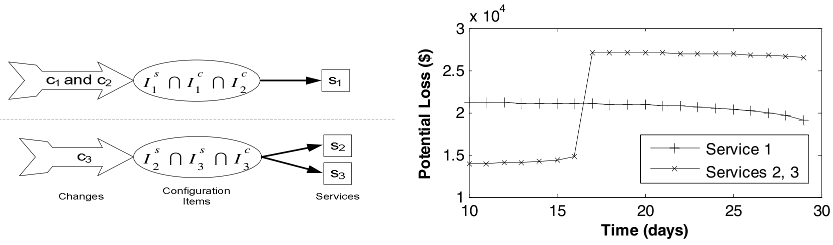


Fig. 3. Expected losses with changes in illustration scenario

6 Related Work

IT management software tools available on the market (such as HP OpenView ServiceDesk and ServiceCenter and BMC Routes-to-Value Change and Configuration Management) provide administrative support to the change management process by tracking a change in all phases of its lifecycle, coordinating its different activities, assigning activities to the appropriate people and monitoring its progress until it is closed. However, these tools provide no support to the decision-making process, and, although concepts such as *risk* and *impact* are present, their definition is rather ambiguous and their assessment is left to the tool user.

CHAMPS [4], a research prototype out of IBM Research, represents the state of the art in automation for change management, but it does not address aspects of project management of the changes such as scheduling activities that require human intervention. Further, it is assumed that business impact is an input parameter and the intended application is to an autonomic computing setting. The linkage model in our work helps to evaluate business impact. Our solution addresses change management challenges holistically: it considers all three components people, process and technology. Thus, our work may be seen as complementary to that of [4]. The work in [1] brings ideas that could be adapted for prioritization/classification of RFCs, since RFCs are frequently related to incidents. Usage of utility functions is particularly attractive.

7 Conclusions and Future Work

From the results of a web survey we carried out in earlier 2006 [7], the main challenges in IT change management were identified as 1) planning/scheduling changes, 2) high number of emergency changes and 3) ill-definition or wrong scoping of requests for change. In this paper we have begun to address the problem of planning and scheduling changes by taking business considerations into account, following a business-driven IT management (BDIM, [6]) approach. We have sketched a reference architecture that follows BDIM principles; the architecture includes a model linking IT availability metrics to business objectives (in our example: minimizing financial loss due service unavailability and SLA violations). A numerical illustration was presented to show how the derived metrics may support change management decisions in order to plan and schedule changes to minimize adverse business impact.

This initial work supports the conception of an automated tool for decision support for planning and scheduling changes. We have received encouraging feedback from the respondents of our survey in [7], to whom we presented the scenario here exposed in a follow up interaction. The respondents agree that the information these metrics provide will definitely “add value to the decision process”. However, much remains to be done before we can embody the capabilities here described into a software tool that is complete enough to be of value to change managers in their decision making activities and while negotiating with customers. The main contribution of this paper is the formalization of a sound base for supporting change scheduling and planning. Our next step will be to formally bring in change windows and solve an optimization problem to find the “best” allocation of changes to change windows. Design and implementation of the decision support tool is also the subject of a future phase of our research.

Acknowledgements

This work was developed in collaboration with HP Brazil R&D.

References

1. Bartolini, C., and Sallé, M., “Business Driven Prioritization of Service Incidents”, *In Proceedings of the 15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM* 2004), 15-17 Nov. 2004, Davis, CA, USA, pp 64-75.
2. IT Governance Institute, “Cobit 4th Edition”, 2006, www.isaca.org/cobit.htm
3. IT Infrastructure Library, “ITIL Service Delivery” and “ITIL Service Support”, Office of Government Commerce, UK, 2003.
4. Keller, A., Hellerstein, J., Wolf, J.L., Wu, K. and Krishnan, V., “The CHAMPS System: Change Management with Planning and Scheduling”, *In Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004)*, IEEE Press, April 2004, pp. 395-408.
5. Sauvé, J. P., Marques, F. T., Moura, J. A. B., Sampaio, M. C., Jornada, J., Radziuk, E., Optimal Design of E-Commerce Site Infrastructure from a Business Perspective, In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), Waikoloa, Hawaii. IEEE Computer Society, 2006. v.8. p.178.3 - 178.3
6. Sauvé, J., Moura, A., Sampaio, M., Jornada, J. and Radziuk, E., “An Introductory Overview and Survey of Business–Driven IT Management”, in Proceedings of the 1st IEEE / IFIP International Workshop On Business-Driven IT Management, in conjunction with NOMS 2006, Vancouver, Canada, pp. 1-10.
7. The HP-Bottom Line Project, “IT Change Management Challenges – Results of 2006 Web Survey” Technical Report DSC005-06, Computing Systems Department, Federal University of Campina Grande, Brazil, March 2006. http://www.bottomlineproject.com/bl/_media /techreport/005-2006.pdf
8. Van Bon, J., Chief Editor, “IT Service Management, an introduction based on ITIL”, itSMF Library, Van Haren Publishing, 2004.
9. Abramowitz, M. and Stegun, I. A. (Eds.). "Beta Function" and "Incomplete Beta Function." §6.2 and 6.6 in *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing*. New York: Dover, pp. 258 and 263, 1972.

Using Argumentation Logic for Firewall Policy Specification and Analysis

Arosha K. Bandara[1], Antonis Kakas[2], Emil C. Lupu[1], and Alessandra Russo[1]

[1] Department of Computing, Imperial College London, London SW7 2AZ
[2] Department of Computer Science, University of Cyprus, Cyprus
{bandara, ack, ecl1, ar3}@doc.ic.ac.uk

Abstract. Firewalls are important perimeter security mechanisms that imple-ment an organisation's network security requirements and can be notoriously difficult to configure correctly. Given their widespread use, it is crucial that network administrators have tools to translate their security requirements into firewall configuration rules and ensure that these rules are consistent with each other. In this paper we propose an approach to firewall policy specification and analysis that uses a formal framework for argumentation based preference reasoning. By allowing administrators to define network abstractions (e.g. subnets, protocols etc) security requirements can be specified in a declarative manner using high-level terms. Also it is possible to specify preferences to express the importance of one requirement over another. The use of a formal framework means that the security requirements defined can be automatically analysed for inconsistencies and firewall configurations can be automatically generated. We demonstrate that the technique allows any inconsistency property, including those identified in previous research, to be specified and automatically checked and the use of an argumentation reasoning framework provides administrators with information regarding the causes of the inconsistency.

1 Introduction

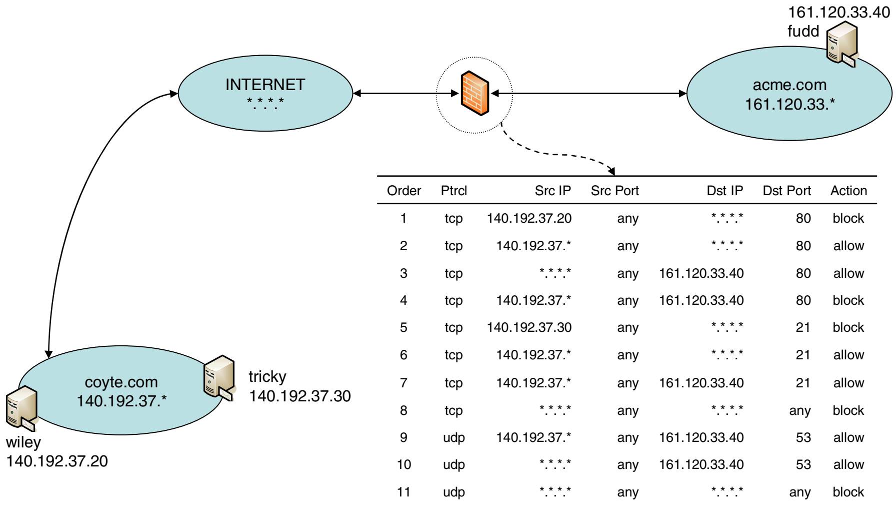
Firewalls are widely used perimeter security mechanisms that filter packets based on a set of configuration rules that are derived from the organisation's network security requirements. The rules are specified in priority order and are of the form:

```
<order> : <action> if <network conditions>
```

where the <network conditions> identify a certain type of traffic, typically from one domain to another under some protocol, and the action field, <action>, typically takes the values "allow" or "deny" thus specifying if the traffic is to be allowed to flow or stopped. The semantics of the firewall policy is given operationally and it is crucially dependent on the total ordering of its rules. The ordering position of a rule is given by a (unique) number in <order> and for a given packet the firewall will check the rules in ascending order. The action field of the first rule whose network conditions are satisfied by the packet determines if it will be allowed or blocked. All subsequent rules are ignored.

In this paper we propose a technique for specifying security requirements within an argumentation based framework for Logic Programming with Priorities (LPP). This allows us to specify and use high-level abstractions for network entities, e.g. protocols, applications, sub-networks. It also allows us to specify relative ordering between security requirements. The framework supports automatic generation of firewall configuration rules that satisfy the requirements including the relative ordering. Additionally, we demonstrate that the framework can detect a range of inconsistencies, including the anomaly types identified by Al-Shaer and Hamed [1], and also perform anomaly resolution.

Figure 1 shows an example system taken from [1] where a firewall is used to protect hosts in an enterprise network (acme.com) from malicious network traffic together with the set of rules that control the behaviour of the firewall. In this example, Rules 8 and 11 implement the default security requirement that all traffic should be blocked unless there is a specific requirement to allow specific types of traffic. These exception cases to the default requirement are implemented by specifying firewall policy rules that have a higher priority ordering than the default policy rule. For example, Rule 7 implements the requirement to *"allow FTP connections from hosts in the coyote.com network to the host ftp.acme.com"* and Rule 1 and 2 to *"allow all HTTP requests from coyote.com to acme.com except those from the host wiley.coyote.com"*.



Order	Ptrcl	Src IP	Src Port	Dst IP	Dst Port	Action
1	tcp	140.192.37.20	any	*.*.*.*	80	block
2	tcp	140.192.37.*	any	*.*.*.*	80	allow
3	tcp	*.*.*.*	any	161.120.33.40	80	allow
4	tcp	140.192.37.*	any	161.120.33.40	80	block
5	tcp	140.192.37.30	any	*.*.*.*	21	block
6	tcp	140.192.37.*	any	*.*.*.*	21	allow
7	tcp	140.192.37.*	any	161.120.33.40	21	allow
8	tcp	*.*.*.*	any	*.*.*.*	any	block
9	udp	140.192.37.*	any	161.120.33.40	53	allow
10	udp	*.*.*.*	any	161.120.33.40	53	allow
11	udp	*.*.*.*	any	*.*.*.*	any	block

Fig. 1. Example network and associated firewall policy rules [1]

In this example, the translation of these requirements into the rules shown is done manually and depends on administrators' knowledge of the low-level network topology and protocols and also having the expertise to assign the correct priority order to the rules. This method of policy specification has the added disadvantage that no link is maintained between the security requirements and the policy rules that implement them. This makes policy specifications hard to understand and it is easy for the administrator to make errors, particularly when dealing with large distributed systems that involve many networks, hosts and applications. Some firewall solution vendors have made an attempt to support high-level abstractions by mapping named traffic classes to low-level properties such as host IP addresses, port numbers and protocols [2]. However, this process involves a significant amount of manual effort on the part of the administrator and the tools provided do not maintain any link between the security requirements and the underlying policy rules.

Another shortcoming with existing approaches to firewall policy specification is the limited support for automated analysis that verifies that the specification satisfies desired security properties and does not contain any inconsistency. Work done by Al-Shaer et al. goes some way to addressing this problem by identifying a number of inconsistency types (or *anomaly types*) and defining an algorithm for detecting the presence of these inconsistencies [1]. Whilst this technique has been extended to detect inconsistency in complex scenarios that involve distributed firewalls, it only detects a fixed set of inconsistency types [3]. Additionally, given that the analysis algorithm operates on the low-level firewall policy rules, it is not able to provide any information about the reasons for an anomaly to exist.

The rest of this paper is organised as follows. In the next section we present information regarding the capabilities of LPP framework together with examples of how the notation can be used to specify network abstractions and security requirements. In section 3 we present the different types of analysis supported by our technique followed by a discussion of our work in section 4. We describe how our work compares with related research in the field in section 5 before presenting our conclusions and plans for further work in section 6.

2 Security Requirements and the Argumentation Framework

One of the objectives of our work is to provide administrators with the ability to specify their security requirements using high-level abstractions that are closer to their natural specifications. We wish to do this in the context of a formal reasoning framework that supports the prioritised ordering of firewall rules and also provides automated analysis capabilities. In this section we present a formal language based on an argumentation framework that is capable of representing background information regarding the network, hosts and traffic types together with network security requirements and relative priorities between rules.

Argumentation has been shown to be a useful framework for formalizing non-monotonic reasoning and other forms of reasoning [4-7]. In general, an argumentation framework is a pair <T,*A*> where T is a theory in some background (monotonic) logic, equipped with an entailment relation, $\models$, and *A* is a binary relation on the subsets of T. These subsets of T form the **arguments** of the framework and *A* is a non-symmetric **attacking relation** between arguments. For any two arguments A1 and A2 we say that **A1 attacks A2** when (A1,A2) belongs to the attacking relation *A*. In this context, an argument A1 attacks A2 if, given the same background knowledge, A1 supports a conclusion that is incompatible with a conclusion supported by A2 and A1 is defined to be stronger than A2. Argumentation reasoning is given through the notion of an *admissible argument*, i.e. an argument that counterattacks another argument. The formal definition of the argumentation framework is presented in [4, 6].

2.1 Representing Security Requirements and Firewall Rules

In the specific argumentation reasoning framework we use in this paper, a theory T is represented in the background logic (L, $\models$), where the language L consists of (extended) logic programming rules of the form:

```
Name: L ← L1, . . . , Ln, (n ≥ 0).
```

Here, L,L_1, . . ., L_n are positive or negative literals. A negative literal is a literal of the form ¬A, where A is an atom. As usual in Logic Programming a rule containing variables is a compact representation of all the ground rules obtained from this under the Hebrand universe. Each ground rule has a unique (parametric) name, *Name*, given at the front of the rule. Using this notation we can specify a security requirement by defining a rule with name `req(…)` that associates a given action with packets that match the source, destination and traffic type. For example, the requirement to *"allow HTTP requests from the coyote.com network to web servers in the acme.com network"* would be defined as follows:

```
req(allow_http_coyote, allow, Pkt):
action(allow, Pkt) ←
  packetFrom(coyote, Pkt), packetTo(Server, Pkt),
  property('web', host, Server), traffic(http, Pkt).
```

The packet terms in the above rule are defined using 5-tuples of the form `pkt(Protocol, SourceIP, SourcePort, DestIP, DestPort)`. In the above definition, the `packetFrom(…)` and `packetTo(…)` predicates are used to map the name of a source or destination entity to the appropriate IP address fields of the packet. In the above definition, the `packetFrom(…)` and `packetTo(…)` predicates are used to map the name of a source or destination entity to the appropriate IP address fields of the packet. These predicates are defined as follows:

```
pktSource(SrcIP):
packetFrom(From, pkt(_,SrcIP, _, _, _)) ← ipaddr(From, SrcIP).
pktDest(DstIP):
packetTo(To, pkt(_, _, _, DstIP, _)) ← ipaddr(To, DstIP).
```

The `ipaddr(…)` predicate is used to define background information regarding the network, namely the IP address of a given network entity. This is described in more detail in the next section.

The overall theory T is separated into two parts: the **basic** part and the **strategy** part. The basic part contains rules (of the form given above) whose conclusions, L, are any literal except the special predicate, `prefer(…)`, which is the only predicate that can appear in the conclusion of rules in the strategy part. Hence rules in the strategy part take the special form

```
Name: prefer(rule1, rule2)← L1, . . . ,Ln, (n ≥ 0).
```

where rule1 and rule2 are the names of any other two rules in the theory. A rule of this form then means that under the conditions L_1, . . . ,Ln, the rule with name, rule1, has priority over the rule with name, rule2. The role of this priority relation is therefore to encode locally the relative strength of (argument) rules in the theory. The priority relation `prefer(…)` is required to be irreflexive. The rules `rule1` and `rule2` can themselves be rules expressing priority between other rules and hence the framework allows higher-order priorities.

We can use the `prefer(…)` predicate to defined security requirements express a precedence relationship between two simpler requirements. For example, the administrator might specify a requirement to *"block any traffic except HTTP requests to fudd.acme.com"*. This type of requirement can be composed from *"block any traffic"* and *"allow HTTP requests to fudd.acme.com"* together with the addition of a rule that makes the latter requirement take precedence. The two simple requirements would be specified as follows:

```
req(block_any, block, Pkt):
action(block, Pkt) ←
  packetFrom(any, Pkt), packetTo(any, Pkt), traffic(any, Pkt).
req(allow_http_fudd, allow, Pkt):
action(allow, Pkt) ←
  packetFrom(any, Pkt), packetTo(fudd, Pkt), traffic(http, Pkt).
```

This is followed by the precedence relationship between these requirements using the `prefer(…)` predicate:

```
   order(allow_http_fudd, block_any):
  prefer(req(allow_http_fudd, allow, Pkt)),
         req(block_any, block, Pkt) ).
```

In addition to representing security requirements, the notation described can be used to specify legacy firewall rules, denoted by the term `fwr(Order, Action, Pkt)`. For example, the first rule shown below represents Rule 9 given in Figure 1. Notice that we use finite domain constraints to specify IP address and port ranges:

```
fwr(9, allow, pkt(udp,ip(140,197,37,D),SP,ip(161,120,33,40), 53)):
action(allow, pkt(udp,ip(140,197,37,D),SP,ip(161,120,33,40), 53))
       ← D in 0..255, SP in 1..65536.
  order(N1, N2):
 prefer(fwr(N1, A1, Pkt), fwr(N2, A2, Pkt)) ← N1 < N2.
```

The second rule shows how we can use the `prefer(…)` predicate to specify the ordering of legacy firewall rules. In this fashion our formal framework can combine the requirements specifications described above with legacy firewall rules.

2.2 Representing Background Information

We can separate out an auxiliary part, T_0, of a given theory, T, from which the other rules can draw background information in order to satisfy some of their conditions. The reasoning of the auxiliary part of a theory is independent of the main argumentation-based preference reasoning of the framework and hence any appropriate logic can be used. In the context of this paper, we can use this feature to specify subnets, hosts and traffic types in a network using the following three predicates:

```
network(Name, [Properties]).
host(Name, [Properties]).
ipaddr(Name, ip(A, B, C, D)).
```

The `network(…)` predicate defines a named network (e.g. acme.com, coyote.com etc) together with a list of associated properties (e.g. wireless, WEP, etc.). Similarly, the `host(…)` predicate defines a host name together with a list of properties associated with that host. Finally, the `ipaddr(…)` predicate associates a particular host (or network) with an IP address (or address range). The `ip(…)` function has four arguments that correspond to each byte of a 32-bit IP address. Using these predicates, the ‘acme.com’ network and ‘fudd.acme.com’ host in the example (Figure 1) would be specified as follows:

```
network(acme, ['acme.com', 'wired']).
ipaddr(acme, ip(161, 120, 33, D)) ← D in 0..255.

host(fudd, ['fudd.acme.com', 'web', 'ftp', 'dns']).
ipaddr(fudd, ip(161, 120, 33, 40)).
```

We use the finite domain constraint 0..255 to specify the range of values for the ‘acme’ network. Notice that each rule in the above definitions is prefixed with a parameterised name, which becomes part of the arguments for the answer derived if the rule forms part of the theory that supports a given goal. For example, if we query the system for the available networks, each answer will be accompanied by an argument set that includes the term `network(…)`, identifying the network rule that defines each network. We can also specify a auxiliary predicate, `property(…)`, which can be used to identify the network elements that have a given property:

```
property(Prop, network, Element)
          ← network(Element, [Props]), member(Prop, Props).

property(Prop, host, Element)
          ←  host(Element, [Props]), member(Prop, Props).
```

In the above definition, the member(Property, Properties) predicate holds if the property denoted by the first parameter is a member of the list denoted by the second. The `property(…)` predicate can also be used to express higher-level, composite properties. For example, the notion that all Linux hosts on a wired network are considered to be secure can be expressed as follows:

```
property(secure, host, Element)
  ←  property(wired, network, Network),
     property(Network, host, Element),
     property(linux, host, Element).
```

Finally we define a predicate, `traffic(Name, Pkt)` that associates the protocol and ports fields of an IP packet with a given type of traffic. This predicate can be also be used to define ranges of ports. For example, we can define the following rules to specify an application called ‘http’ which matches TCP packets from any non-reserved port (1024-65536) to port 80; and a generic application called ‘any’ which matches packets containing any port number and protocol:

```
traffic(http, pkt(Prtcl, SrcIP, SP, DstIP, DP)) ←
  Prtcl=tcp, SP in 1024..65536, DP = 80.

traffic(any, pkt(Prtcl, SrcIP, SP, DstIP, DP))) ←
  (Prtcl=tcp; Prtcl=udp), SP in 1..65536, DP in 1..65536.
```

It is important to note that specifying this background information is a one-time task that can be automated using host/service discovery tools. Of course the specification will have to be updated if there are any changes in the system, but this process can also be automated.

In addition to background information regarding the network, the auxiliary part of our theory also contains the definition of what constitutes a conflict (over and above the standard conflict of classical negation, i.e. between an atom, A and its negation

¬A). This is given through the definition of an auxiliary predicate, `complement(…)`, which is of the form:

```
    complement(L1, L2) ← B.
```

stating that literals L1 and L2 are conflicting under some (auxiliary) conditions B. Typically, the conditions B are empty and the definition of the `complement(…)` predicate is kept simple. Also we will assume that the conditions of any rule in the theory do not refer to the predicate `prefer(…)` thus avoiding self-reference problems. Note also that the definition of `complement(…)` always includes that any ground atom, `prefer(rule1, rule2)`, is incompatible with the atom `prefer (rule2, rule1)` and vice-versa. In the context of firewall policies, we would define the actions *allow* and *block* to be complementary using the following rule:

```
    complement(action(allow, _), action(block, _)).
```

The logical framework for argumentation and preference reasoning described here has been realised in the GORGIAS tool developed at the University of Cyprus [8] and has been used to implement the examples and generate the results presented in this paper. This tool provides a query, `prove([L1, L2, …, Ln], Args)`, which generates the set of admissible arguments, Args, that support the conjunction of terms $L_1, \ldots, L_n$ for a given theory. In order to support the analysis of security requirements and firewall policies, we define the following auxiliary query to determine if a particular packet will be allowed or blocked by a firewall together with the rule (or requirement) that causes this decision and the supporting arguments:

```
    packet_action(Action, Pkt, Rule, Args) ←
      prove([action(Action, Pkt)], Args), member(Rule, Args),
      (Rule=requirement(R, Action, Pkt); Rule=fwr(N, Action, Pkt)).
```

3 Analysing Firewall Policies

As a network grows, the task of managing the network security policies quickly becomes unwieldy. Therefore it is very important to provide administrators with support to analyse the policy specification and ensure that desired properties hold. These analysis tasks can be divided into the following categories:

1. **Anomaly Detection:** Analysing the policy specification for potential anomalies.
2. **Property Checking:** Performing "what-if" analysis to determine if a given class of traffic will be forwarded or blocked. For example, *"Which packets are allowed to reach the host fudd.acme.com?"* This type of query can also be used to verify that the policy specification satisfies desired behaviour.
3. **Anomaly Resolution:** Determining the correct ordering of policy rules to ensure that anomalies are avoided (i.e. ensuring that rules related to exception cases are given a higher precedence than general rules).

3.1 Anomaly Detection

Al-Shaer et al [1] have identified four firewall policy anomaly types – shadowing, generalisation, correlation and redundancy and here we show how these anomalies can be detected using the argumentation logic framework. From the description of the various anomaly types it is clear that the key determinants of an anomaly is whether the packets that match a rule are a subset (or superset) of the packets matched by another rule; and the relative ordering of the rules. For example, rule R2 is said to be shadowed by R1 if the rules specify incompatible actions, R1 has preference over R2 and every packet that matches R2 is matched by R1. In order to detect this type of anomaly we define the following rule:

```
anomaly(shadow, R1, R2, Pkt1)←
  packet_action(A1, Pkt1, R1, _),
  complement(action(A1,_,_), action(A2,_,_)),
  packet_action(A2, Pkt2, R2, _),
  match(subset, Pkt1, Pkt2).
```

The above rule identifies a requirement R1 where the matching packets, Pkt1, are a subset of the packets that match another requirement R2 and R2 defines an incompatible action. The preference reasoning capabilities of the LPP framework ensures that the above query identifies rules derived from R2 that have higher precedence than rules derived from R1. Rules that participate in a generalisation anomaly would cause a shadow anomaly if their relative order was reversed. We use this property to define the following rule to detect this type of anomaly:

```
anomaly(generalisation, R1, R2, Pkt2)←
  packet_action(A1, Pkt1, R1, _),
  complement(action(A1,_,_), action(A2,_,_)),
  packet_action(A2, Pkt2, R2, _),
  match(subset, Pkt2, Pkt1).
```

The above definition identifies a policy rule derived from requirement R2 that takes precedence over a rule derived from requirement R1, where the packets matched by R2 are a subset of those matched by R1 and the actions of R1 and R2 are complementary.

Correlation anomalies occur when two rules with complementary actions match the same packets, and the rules are not part of a shadowing or generalisation anomaly. These can be detected using the following rule:

```
anomaly(correlation, R1, R2, Pkt)←
  packet_action(A1, Pkt, R1, _),
  complement(action(A1,_,_), action(A2,_,_)),
  packet_action(A2, Pkt, R2, _),
  ¬ anomaly(generalisation, R1, R2, _),
  ¬ anomaly(generalisation, R2, R1, _),
  ¬ anomaly(shadow, R1, R2, _).
```

Redundancy anomalies differ from the other types in that they involve rules that specify the same action. We define the following rule to detect this type of anomaly:

```
anomaly(redundant, R1, R2, Pkt1)←
  packet_action(A, Pkt1, R1, _), packet_action(A, Pkt2, R2, _),
  R1 \== R2, match(subset, Pkt1, Pkt2).
```

Using these rules, we can detect all the anomalies in a specification using a single high-level query. For example, performing such a query on the example system shown in Figure 1 would generate the following result:

```
?- findall(Type-(R1, R2), anomaly(Type, R1, R2, _), List).
List =  shadow-(deny_coyote_http_fudd,allow_coyote_http)
        shadow-(deny_coyote_http_fudd,allow_http_fudd)
        generalise-(deny_wiley_http,allow_coyote_http)
                ...
```

```
generalise-(allow_udpdns_fudd,deny_all)
correlated-(deny_wiley_http,allow_http_fudd)
correlated-(deny_tricky_ftp,allow_coyote_ftp_fudd)
redundant- (allow_coyote_ftp_fudd, allow_coyote_ftp)
redundant- (allow_coyote_udpdns_fudd,allow_udpdns_fudd)
```

3.2 Property Checking

In addition to checking for the anomaly types identified in the literature, the formal framework for firewall policy specification described in this paper is a general one that can be used to check if a specification satisfies other properties. For example, the administrator might wish to verify which packets are allowed to reach the host fudd.acme.com. This property would be checked by the following high-level query:

```
?- packet_action(allow, Pkt, Rule, Args), packetTo(fudd, Pkt).

      Rule = allow_coyote_http
    Packet = pkt(tcp, ip(140,192,37,D1), SP, ip(161,120,33,40), 80)
        D1 = 0..255, SP = 1024..65536
Arguments:
requirement(allow_coyote_http, allow, pkt(tcp, coyote, SP, any, 80)).
pktDst(any,ip(161,120,33,40)).
pktSrc(coyote,ip(140,192,37,D1)).
...
```

The arguments explain that a TCP packet from 140.192.37.*-port:1024-65536 to 161.120.33.40-port:80 is allowed because the requirement 'allow_coyote_http_allow' specifies that packets from the 'coyote.com' network to port 80 of any host should be allowed. Furthermore, the arguments show how the IP address and port ranges in the allowed packet match the IP addresses of 'coyote.com' and 'fudd'.

Notice that the use of the finite domain constraints for IP address and port ranges means that the query returns an expression that describes all the packets that are allowed to reach the host 'fudd'. The ability to consider the relative priorities between security requirements and also provide this type of coverage of the potential packet space when reporting results is possible because we are using a logic programming based approach that supports preference reasoning.

3.3 Anomaly Resolution

Of the anomaly types defined in the previous section, only redundancies and shadowing anomalies are considered to be errors. Of these, shadowing anomalies can be resolved by reversing the relative ordering of the two rules. This can be expressed in our framework using a 'higher-order' preference reasoning rule as follows:

```
resolve(shadow, R1, R2):
prefer(R1, R2) ← anomaly(shadow, R1, R2, _).
```

The above rule states that preference should be given to rule R1 over R2, i.e. the shadowed rule is given higher priority. Redundancy anomalies on the other hand can be resolved by ensuring the redundant rule has lower priority. This resolution process is specified in our formal framework as follows:

```
resolve(redundancy, R1, R2):
prefer(R2, R1) ← anomaly(redundant, R1, R2, _).
```

Here the `anomaly(…)` predicate holds if R1 is redundant to R2 and the `prefer(…)` predicate defines that R2 should take precedence over R1. In our framework, performing the resolution actions shown above will remove any redundancy and shadowing anomalies from the specification. Additionally, the decision to perform a particular resolution action will be explained with a set of arguments.

4 Discussion

In the study of the analysis of firewall policies we have shown specifically that the various types of anomalies in firewall policies, identified separately in the literature, can be captured naturally under the same and unified definition based on the standard notion of an admissible argument in Logic Programming with Priorities (LPP). This high level definition means (a) that we are more complete in capturing the notion of anomaly and (b) that our definitions remain invariant as we further develop the types of policy supported by the notation, e.g. as we consider extensions of policies for distributed firewalls. The high-level of expressivity of the LPP framework, particularly its ability to represent preference orderings which can be conditional on some background properties means that the formalism can accurately capture the behaviour of a firewall where policies are specified with an explicit priority order. The LPP framework can be used to detect all the anomaly types identified in the literature and also supports other types of property checking, thus allowing an administrator to verify the behaviour of a firewall that is controlled by a given set of requirements. Whilst we have yet to complete experiments on large policy sets, the complexity of the argumentation reasoning framework for the restricted type of theory described in this paper has been shown to be P-complete [9]. We are working to validate the scalability of our approach as part of our ongoing research efforts.

In addition to experimenting with larger policy sets, we also hope to work on more complex scenarios involving multiple firewalls in the network. In such a system, where policies will be distributed across the network the problem of the existence of anomalies is more severe as there are more possibilities for conflicts to occur. We can have situations where one component decides to accept traffic whereas another component decides to deny it. For example, an upstream firewall blocking a traffic that is permitted by a downstream firewall is a type of inter-firewall shadowing anomaly. In a “classical” approach to anomaly detection the definition of this anomaly requires a detailed (and somewhat ad hoc) examination of the pairs of rules from the two firewalls. In our declarative approach this anomaly falls under the same definition given above.

5 Related Work

Work presented by Wool et al., proposes a high-level language for specifying network information and firewall policies that allows firewall configuration to be performed at an abstraction level that is closer to high-level programming. This work has led to the development of a number of tools that support offline firewall policy analysis and management [10]. However, the analysis process does not detect specific anomaly types such as shadowing and redundancy.

Uribe and Cheung have developed a technique for automating the analysis of firewall and network intrusion detection systems that uses constraint logic programming to model the networks and policies [11]. The use of finite domain constraints to specify IP address and port ranges means that the analysis process covers all IP address and port combinations for potential problems. However, the technique does not support specification of explicit priorities between firewall policy rules and the tool does not provide administrators with any explanation to support the analysis results generated.

Al Shaer et al. and Yuan et al., have focussed on tools and techniques for analysing legacy firewall policies for networks with centralised and distributed firewalls [1, 3, 12]. We use the classification of anomalies into the types: shadow, correlation, generalisation and redundancy anomalies presented in [1] to specify the analysis rules used in the framework presented in this paper. One shortcoming of their approach is the dependence on legacy firewall policies in order to perform anomaly detection and resolution. In contrast, our approach allows network security requirements to be specified using high-level notations whilst still being capable of a range of analysis tasks such as anomaly detection, resolution and property checking. Additionally by using an argumentation reasoning framework, our approach has the advantage that the administrator is given an explanation of the analysis results and resolution actions.

6 Conclusions and Future Work

We have presented an approach to specifying network security requirements that is based on Argumentation for Logic Programming with Priorities (LPP). The use of logic programming allows the specification to include high-level abstractions such as networks, hosts, traffic types and their associated properties. This means that administrators can specify their network security requirements in more familiar terms, without having to know the exact IP address and port ranges for a given traffic flow. We have shown that the technique is capable of performing a range of analysis tasks, from detecting the firewall anomaly types identified in the literature to performing more general property checking and conflict resolution. The use of LPP allows preferences to be encoded, thus allowing complex reasoning over the relative priorities between rules. Additionally, the encoded preferences can be conditional on arbitrary system properties, an approach that allows greater flexibility than simple assigned priorities between rules. Also, because LPP is implemented using argumentation reasoning, the results of performing queries are enhanced by explanations containing the rules that support a particular conclusion. This information is particularly helpful to the user in understanding the reason for a traffic flow to be allowed or blocked by the firewall. The current implementation of the technique presented in this paper focuses on security requirements specification for firewalls. However, given an appropriate formalisation of the underlying system, the use of LPP can be extended to other application domains, such as network QoS management.

Our system is implemented using the GORGIAS tool running in a standard Prolog environment. Given a formal description of the network elements and security requirements, it provides support for checking general properties, including checking for the presence of the anomaly types identified in the literature, and also supports anomaly resolution. At present we are focused on extending the tool to provide automated generation of ‘anomaly-free’ operational firewall policies. Additionally are developing a GUI that will shield the administrator from the underlying formal notation, providing an interface that simplifies the process of defining their network security requirements and analysing them for consistency. Our future work also includes extending the formal notation to include information required to specify and analyse network security requirements that are implemented using distributed firewalls.

Acknowledgements

We acknowledge financial support for this work from the EPSRC (Grant Numbers - GR/R31409/01, GR/S79985/01 and GR/T29246/01) and IBM Research.

References

[1] E. S. Al-Shaer and H. H. Hamed. "Firewall Policy Advisor for Anomaly Doscovery and Rule Editing." In Proceedings of 8th IFIP/IEEE International Symposium on Integrated Network Management, Colarado Springs, CO, IEEE, March 2003.
[2] Cisco. "Cisco PIX Firewall Configuration White Paper (DOCID: 68815), http://www.cisco.com/warp/public/707/ezvpn-asa-svr-871-rem.pdf", Cisco Inc, 2006.
[3] E. S. Al-Shaer and H. H. Hamed. "Discovery of Policy Anomalies in Distributed Firewalls." In Proceedings of 23rd IEEE Communications Society Conference (INFOCOM), Hong Kong, IEEE, March 2004.
[4] P. M. Dung (1995). "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games." Artificial Intelligence(77): 321-357, 1995.
[5] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni (1997). "An abstract argumentation theoretic approach to default reasoning." Artificial Intelligence 93: 63-101, 1997.
[6] A. Kakas, P. Mancerella, and P. M. Dung. "The acceptability semantics for logic programs." In Proceedings of 11th International Conference on Logic Programming, Santa Marherita Ligure, Italy, 1994.
[7] H. Prakken and G. Sartor. "A system for defeasible argumentation, with defeasible priorities." In Proceedings of International Conference on Formal and Applied Practical Reasoning, Springer-Verlag, LNAI 1085, 1996.
[8] Gorgias. "Argumentation and Abduction, http://www2.cs.ucy.ac.cy/~nkd/gorgias/",
[9] Y. Dimopoulos, B. Nebel, and F. Toni (2002). "On the Computational Complexity of Assumption-based Argumentation for Default Reasoning." Artificial Intelligence 141: 57-78, 2002.
[10] A. Mayer, A. Wool, and E. Ziskind (2006). "Offline firewall analysis." International Journal on Information Security 5(3): 125-144, 2006.
[11] T. E. Uribe and S. Cheung. "Automatic Analysis of Firewall and Network Intrusion Detection System Configurations." In Proceedings of ACM Workshop on Formal Methods in Security Engineering, Washington, DC, ACM Press, October 2004.
[12] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra. "FIREMAN: a toolkit for FIREwall Modeling and ANalysis." In Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, May 2006.

ZERO-Conflict: A Grouping-Based Approach for Automatic Generation of IPSec/VPN Security Policies*

Kuong-Ho Chen, Yuan-Siao Liu, Tzong-Jye Liu, and Chyi-Ren Dow

Department of Computer Science, Feng Chia University,
No. 100 Wenhwa Rd., Seatwen, Taichung, Taiwan 40724, R.O.C
cyne@pluto.iecs.fcu.edu.tw, {m9301324, tjliu, crdow}@fcu.edu.tw

Abstract. IPSec/VPN management is a complicated challenge, since IPSec functions correctly only if its security policies satisfy all administrated requirements. Computer-generated security policies tend to conflict with each other, which would causes network congestion or creates security vulnerability. Thus conflict resolving has become an issue. In this paper, a method to automatically generate policies is proposed. Instead of performing complicated conflict-checking procedures as most existing works do, the proposed *Zero-Conflict* algorithm is able to predict and avoid conflict in advance by using *requirement groups* and *cut points* techniques. Since policies are established without the need to perform backward conflict check, thus yielding a significantly less time-complexity, which is *O(nlogn)*. Experimental results show that it maintains a satisfactorily minimal numbers of generated tunnels.

Keywords: IP security, network management, policy conflict, security policy, security requirement.

1 Introduction

Network management in large distributed networks, in particular IPSec/VPN management [8, 13], is a complicated challenge. IPSec functions will be executed correctly only if policies are correctly specified and configured, but due to the growing number of secure Internet applications today, IPSec policy [1, 7] deployment has become rather complex in large distributed networks, and manual configuration is rather tedious, ineffective, and often erroneous. On the other hand, a policy-based management system treats network requirements as goals to be achieved, automatically translates them into low-level machine-understandable policies, and systematically applies them to right network devices. Since IPSec is basically a typical policy-enabled networking service, policy-based network management [3, 11] is a good solution in handling complicated IPSec policy, and various solutions for IPSec/VPN policy management have been proposed in researches such as [2, 4, 5, 6, 12, 13].

A class of high level policy is defined in [5], which is called *security requirement.* Conceptually, security requirements (high level policy) are like goals, while implemental

* This Research is supported in part by the National Science Council under the grants No. NSC NSC94-2213-E035-025, NSC95-2221-E035-071.

IPSec policies (low level policy) are like specific plans to achieve these objectives. IPSec policies are considered correct only if these policies as a whole are able to satisfy all specified security requirements. However, security requirement and IPSec policy may not directly map to each other since one security requirement might be satisfied by several sets of implemental policies. Moreover, it is possible that there are conflicts between requirements and policies. If such conflicts exist on one of the gateways/routers, packets could be dropped, network could be down, and security could be breached. Conflicts occur when given requirements conflict with each other, or when a set of policies binding together is unable to support given requirements. An exemplary scenario of the latter case is given in Table 1 and Fig. 1. (In this paper, the terms *policy* and *tunnel* will be used interchangeably.)

Table 1. Two Requirements

	Requirement
Req_1	All traffics from A to D must be applied encryption
Req_2	All traffics from C to E must be applied authentication

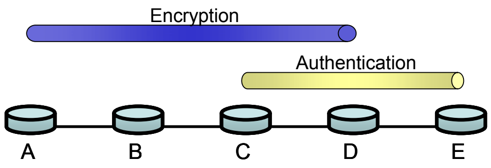


Fig. 1. Overlapping Tunnels

In this scenario, there are two requirements for all traffics from A to E: the coverage of Req_1 is from A to D and the coverage of Req_2 is from C to E. According to these two requirements, two tunnels were built: one from A to D with encryption and another from C to E with authentication. With these tunnels, all packets are encapsulated in A, encapsulated again in C, and then sent to E. E decapsulates these packets and finds that their destinations are D, thus send them there. Finally, D decapsulates them and sends the packets to their final destination E. However, while the original requirement was to authenticate the traffic from C to E, the traffic is actually sent without protection from D to E due to tunnel overlapping.

Thus said, in spite of policy generation, an IPSec policy management system will also need to tack tunnel overlapping and identify possible policy conflicts. Researches so far in automatic IPSec policy generation [2, 4, 6, 12] focus their efforts on conflict resolving: in these algorithms, each newly generated security policy is compared with existing policies to check for conflict. Once found, operations are called for conflict resolve. The process of requirement comparison, however, is rather time-consuming, since any change or addition in security requirements will require the entire execution of conflict-check (and possibly conflict-resolving) procedure.

If tunnels were constructed in a way such that conflicts are predicted and avoided in advance, establishment of policies without need for time-consuming backward conflict check would be made possible, thus yielding a faster result. With this in mind, this paper proposed a Zero-Conflict algorithm, which is an efficient conflict-avoiding method for automatic generation of IPSec/VPN security policy. In this paper, a 3-phased automatic policy construction procedure is proposed, which seeks to lower the complexity of conflict dealing by dividing requirements into groups, and establish *bus tunnels* and *branch tunnels* inside each group. In comparison with several existing methods [2, 4, 12], which mostly came with the efforts of $O(n^2)$, this approach requires the effort of only $O(n\log n)$, where n is the number of requirements. Simulation results also show that the proposed Zero-Conflict approach maintains an appropriately minimal number of established tunnels.

The rest of this paper is organized as follows. Related works are addressed in Section 2. Analysis of policy conflict problem is described in Section 3. Automatic policy generation algorithms are described in Section 4. The complexity analysis and simulation are in Section 5 and Section 6. Finally, conclusions are made in Section 7.

2 Related Works

In this section, research backgrounds and literatures related to our work are described in Section 2.1, including the categories and the definitions of security requirements. Various approaches for automatic IPSec/VPN policy generation are then described and discussed in Section 2.2, including bundle approach [4], direct approach [4], Order-Split approach [12], and Conflict-Free approach [2].

2.1 Security Requirements

In [5], two levels of security policies are defined: the *requirement* level and the *implementation* level. The needs to distinguish high-level security requirements and low-level policies were addressed in [9, 10]. A security policy set is correct if and only if it satisfies all the requirements. A requirement R is a rule of the following form: If condition C then action A:

$$R \equiv C \rightarrow A \tag{1}$$

There are four cases of requirements defined in [5]:

- Access Control Requirement (ACR):
  flow id → deny | allow
- Security Coverage Requirement (SCR):
  flow id → enforce (sec-function, strength, from, to, [trusted-nodes])
- Content Access Requirement (CAR):
  flow id, [sec-function, access-nodes] → deny | allow
- Security Association Requirement (SAR):
  flow id, [SA-peer1, SA-peer2] → deny | allow

flow id is used to identify a traffic flow, and is composed of 5 to 6 sub-selectors including *src-addr*, *dst-addr*, *src-port*, *dst-port*, *protocol*, and optional *user-id*. A requirement is satisfied if and only if all packets selected by the condition part execute the action part of the requirement. As were mentioned in [2, 12], other requirements such as SAR or CAR can be validated after SCR results are produced.

ACR policies also can be determined after tunnel configurations are done. Therefore the algorithm in this paper will seek to focus on the handling SCR requirements only.

2.2 Previous Works

Bundle approach [4] is the first algorithm for automatic policy generation. In this approach, the problem is divided into two phases. From given requirements, the entire traffic is first divided into several disjointed traffic flows, which are called bundles. Sets of security policies are then built from each bundle. Although correct and solution-guaranteed, this approach is not efficient since redundant tunnels for the same area could be built from different bundles.

Direct approach which was also proposed in [4], tunnels are built from each requirement directly, and all the while with the system making sure new tunnels do not overlap with any existing ones. If overlapping occurs, the new tunnel is divided into two connecting tunnels. In comparison with bundle approach, this approach produces fewer tunnels and has better efficiency. It does not, however, yield solutions for every case.

Ordered-Split algorithm [12] is based on traditional task-scheduling schemes for automatic policy generation. Original requirements are converted into tie-free requirement sets; a minimal sized Canonical Solution for the new requirements are then acquired. The condition for a Canonical Solution is that no two tunnels share the same start as well end, while the condition for a tie-free requirement is that no two requirements share the same *from* and *to*. According to [12], this algorithm generates fewer tunnels than Bundle/Direct approach, and is free of tunnel-redundancy problem. Its time-complexity is $O(n^2)$.

Conflict-Free approach [2] focuses on the handling with the intersection relationship between tunnels. In this approach all tunnel are made as long as possible since if two security policy sets have the same number of tunnels, the set which has longer average tunnel length will be preferred since longer tunnel decreases the number of times a traffic has to be encapsulated/decapsulated. The time-complexity, of this algorithm is $O(n^2)$.

3 Analysis of Overlapping Relationship Possibilities

A policy conflict is caused when two or more tunnels have certain overlapping relationships. To be more specific, when packets in one tunnel are passing through a node in the network, they will be pulled into other tunnels due to the policies of the same node, which is likely to cause a policy conflict. To better understand the nature of policy conflict, shown in Fig. 2 are the six possibilities of overlapping relationships between two tunnels, whose analysis could be used to find the possible cause of conflicts.

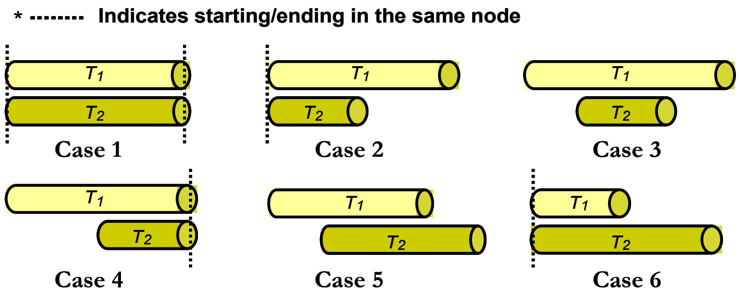


Fig. 2. Overlapping Relationship Possibilities for Two-tunnel Scenario

According to [2], conflicts could possibly appear in case 5 and case 6 only. In case 5, packets are encapsulated at the start of T_1. When traveling through the network to the middle node, which is the start of T_2, they will be encapsulated again, then be directly sent to the most right node, which is the end of T_2, and be unwrapped and sent back to the end of T_1. After arriving at the end of T_1, the traffic will leave T_1 and be sent to the most right node. Since there is a sent back occurrence, conflicts are likely to be caused. In case 6, T_1 and T_2 start at the same node, and packets will be encapsulated twice here. When being unwrapped first time at the end of T_2, these packets will be sent back to the end of T_1, which might cause a policy conflict. Note that case 2 and case 6 differ only in the order of input tunnels. While processing, algorithms in [2, 4, 6, 12] will perform extra order switching in order to convert case 6 into case 2, thus avoiding conflict, which generates extra overhead. In contrast with these, we hope to construct a scheme that is free of this problem of ordering, whereas policies are processed in the order as they were inputted.

To sum it up, a conflict exists when *send back* occurs in two overlapping tunnels, therefore only case 5 and case 6 can possibly cause conflicts. Knowing this in advance, our algorithm, different from those aforementioned, seeks to avoid the occurrences of these two situations at all, rather than dealing with them headfirst.

4 Zero-Conflict Algorithm

Taking advantages from analysis above, *Zero-Conflict* algorithm was designed with the concepts of *requirement group* and *cut point* in mind. In this section these two major concepts are described, the mechanism of the Zero-Conflict algorithm itself explained, and an example is also given for better understanding.

4.1 Requirement Group

In a two-tunnel scenario, if these two tunnels do not overlap with each other, no conflict will occur. A requirement group is a set of requirements that do not overlap with the requirements belonging to other group. In other words, a group is composed of at least one or more overlapping requirements so that the conflicts will appear only in their own respective groups.

4.2 Cut-Point

Once requirement groups are finalized, conflicts inside each group are to be resolved. Common methods for resolving overlapping is to divide the requirement in question into two non-conflicting ones.

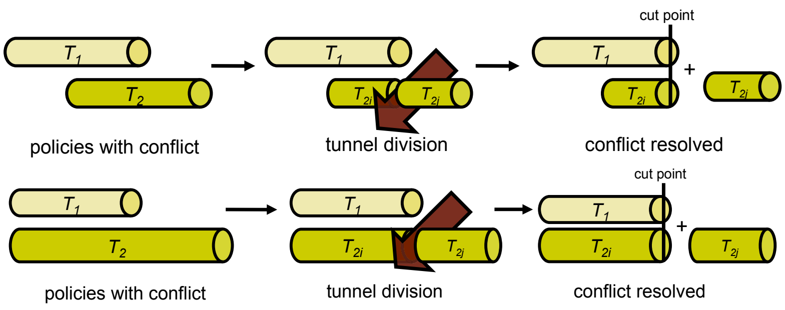


Fig. 3. Conflict Resolving with Tunnel Division

The center of the problem is to find where to "cut", thus the *cut point*. An observation was made: If the original requirement list is first sorted by *from* values in ascending order and tunnels are established accordingly, when conflicting cases in Fig. 3 appear, the tunnel T_2 will be divided at the end of the tunnel T_1 (thus $T_2.cutpoint=T_1.to$). Thus T_2 will be replaced by $T_{2i}=(T_2.from, T_1.to)$ and $T_{2j}=(T_1.to, T_2.to)$. Any subsequent tunnels, if in conflict with T_1, will be divided at the same cut point, thus $T_1.to$.

Thus if every *to* in the requirement list is treated as a cut point, and all tunnels are to be divided according to these cut points, conflicts can be avoided (In this way, a tunnel covering *n* cut points will be divided *n* times). Basing on this assumption, two facts can be derived: a) a cut point is the end of one tunnel and the start of another, but a start of one tunnel is not necessarily a cut point. b) Between every two neighboring cut points in a group, there must be at least one tunnel. According to a) and b), those tunnels whose establishments are guaranteed can be determined in early stage of the algorithm. These are called *bus tunnels*, which will be established after the acquisition of the *to* of a security requirement *in advance*, and serve as backbones shared by all requirements in the same group. Branching tunnels will later be built from these buses, covering remaining areas, which are henceforth called *branch tunnels*. It could be observed that the *from* and *to* of a bus tunnel are both cut points. For a branch tunnel, its *from* must not be a cut point, while its *to* must be one. Branch tunnels, in conjunction with bus tunnels, satisfy the overall covering demands of the requirement list. Therefore any given requirement can satisfied by connecting its *from* with a closest bus tunnel using a branch tunnel.

4.3 Zero-Conflict Algorithm

Taking advantages from analysis above, an algorithm for automatic policy generation which avoids the two conflict cases can thus be designed, which is called "*Zero-Conflict* algorithm". The pseudo code of Zero-Conflict is shown in Fig. 4. Sub- functions are explained as follows:

```
Zero-Conflict Algorithm
Zero-Conflict_Algorithm (Reqs)
{
    remove_length_1_requirement (Reqs, lenegth1_Req_list);
    sort_by_from_value (Reqs);
    assign_group_number_to_each_requirement (Reqs);
    gather_cut_points_for_each_requirement_group (Reqs, Cut-Point_list);

    build_bus_tunnel (Cut-Point_list, Policy_List);
    build_branch_tunnel (Reqs, Cut-Point_list, Policy_List);
    build_length_1_tunnel (length1_Req_list, Policy_List);

    remove_redundant_tunnel (Policy_List);

    return Policy_List;
}
```

Fig. 4. Pseudo code for Zero-Conflict Algorithm

remove_length_1_requirment (Reqs, length1_Req_list). Requirements with their *from* and *to* as neighbors (hop count is 1, thus one-hop requirement) does not conflict with other requirements, but increases the number of cut points unnecessarily, thus has to be moved to backup space *length1_Req_list* and be processed in later stage.

sort_by_from_value (Reqs). The original requirement list is then sorted by *from* in ascending order. Note that most subsequent operations are done directly on the sorted requirements list, thus lowering their time-complexity to *O(n)*.

assign_group_number_to_each_requirement (Reqs). To the sorted requirement list, a variable *max_end_node* is used to record the end node of current group, which is also the *to* of the first requirement. A single *n*-loop operation is executed to determine the group of each requirement. If a requirement belongs to current group, its *from* must be less or equal to *max_end_node*, else it belongs to the next requirement group. In the latter case, a new group is created, and *max_end_node* is set to the *to* of first requirement in this group. Note that if *to* is greater then *max_end_node*, then *max_end_node* is set to the *to.*

gather_cut_points_for_each_requirement_group (Reqs, Cut-Point_list). The end nodes from each requirements are collected in non-repeated fashion as cut points, and then sorted with their group numbers as primary key and *to* as second key, thus generating the cut point list.

build_bus_tunnel (Cut-Point_list, Policy_List). For every two neighboring cut points in each requirement group, bus tunnels are established between them.

build_branch_tunnel (Reqs, Cut-Point_list, Policy_List). Once bus tunnels are established, a single *n*-loop operation is executed to establish branch tunnels. Since the *to* of a branch tunnel is a cut point, which itself is the *from* of a certain bus tunnel. Thus for each requirement, a branch tunnel is established between its *from* and the nearest cut point, thus linking itself with the backbone of the requirement group. Note that for each *from*, only one branch tunnel will be established, since there may be multiple requirements with identical *from*.

build_length_1_tunnel (length1_Req_list, Policy_List). Finally, the removed one-hop requirements are established. (After this function is completed, established tunnels are already able to satisfy all requirements.)

remove_redundant_tunnels (Policy_List). This function removes redundant tunnels. If the area covered by several tunnels is also the covered area by a single tunnel, the latter tunnel is considered redundant, and will be removed.

Generated tunnels are first sorted by *from* in descending order and by hop count (hop count = *to-from)* in ascending order. This is due to that all tunnels excluding one-hop end in a cut point. Since hop count is the distance between *to* and *from*, therefore sorting by *from* equals to sorting by the distance between the start of each tunnel and its nearest cut point. In this way, shorter tunnels will be pushed toward the top. Since a redundant tunnel can only be replaced by several shorter tunnels (thus tunnels with less hop count) that interconnect together, therefore a variable M_a is used to record the area covered by current set of connecting tunnels.

An *n*-loop operation is then executed for removal. Since there could be multiple tunnels connecting together, a variable M_a is used to record the area covered by current set of connecting tunnels. The initial value of M_a is set to the area covered by the first tunnel, thus M_a*=(*T_o*.from,* T_0*.to).*

Each tunnel T_i is first compared with M_a. If identical, T_i will be erased. If not, the operation proceeds to see whether T_i is connected with the area covered by M_a. If T_i*.to* = M_a*.form*, then T_i is added into the set (thus M_a*.from* = T_i*.from)*. Else, M_a is set to (T_i*.from,* T_i*.to*), and then the loop is carried onto the next tunnel.

4.4 An Example of Zero-Conflict Method

For better understanding, an example is given in Table 2, where 8 requirements are input.

First of all, one-hop requirements are to be removed. In this case, Req_3 is removed, and the remaining requirements are sorted by *from*, thus generating Table 3.

To the finding of group numbers, there are two groups in this case. The first group, $G_0=\{Req_2, Req_4, Req_5, Req_6, Req_8\}$, with *max_end_node* = 6. While processing Req_1 it can be noted that Req_1*.from* is greater than current *max_end_node*, thus forming a new group, $G_1=\{Req_1, Req_7\}$, with *max_end_node = 10.*

Subsequently, cut points in each requirement groups are to be decided. In this case, they are *{5, 6}* for G_o, and *{9, 10}* for G_1.

Thus onto the construction of bus tunnels. In this case, $T_1=(5,6)$ is established for G_0, while $T_2=(9,10)$ is established for G_1.

For branch tunnel construction, the nearest cut point for requirements in G_0 is 5. And since Req_5 and Req_8 share the same *from*, only one branch tunnel will be generated for these two requirements. Thus 4 branch tunnels : $T_3= (1,5)$, $T_4= (2,5)$, $T_5= (3,5)$, $T_6= (4,5)$ are established for G_0, and 2 for G_1 : $T_7= (7,9)$, $T_8= (8,9)$.

Now the one-hop requirements removed earlier can be put back and established, thus T_9= *(2,3)*.

Onto the redundancy check. Sorted by *from* in descending order, the check will start from the farest tunnel, which is T_2, thus M_a=*(9, 10)*.

Table 2. An Example of Eight Requirements

Requirement	
Req_1	SCR(E, 7, 9)
Req_2	SCR(E, 1, 5)
Req_3	SCR (A, 2, 3)
Req_4	SCR (E, 2, 6)
Req_5	SCR (E, 3, 5)
Req_6	SCR (E, 4, 6)
Req_7	SCR (E, 8, 10)
Req_8	SCR (A, 3, 6)

Table 3. Sorted Requirement List

Sorted Requirement	
Req_2	SCR (E, 1, 5)
Req_4	SCR (E, 2, 6)
Req_5	SCR (E, 3, 5)
Req_8	SCR (A, 3, 6)
Req_6	SCR (E, 4, 6)
Req_1	SCR (E, 7, 9)
Req_7	SCR (E, 8, 10)

T_8 is then compared with M_a, and it is found that M_a*.from* = T_8*.to*, indicating that T_8 are connected to current M_a, therefore T_8 is merged with M_a, thus M_a = *(*T_8*.from,* M_a*.to)*.

T_7 is then compared with M_a, and it is found that M_a*.from* ≠ T_7*.to*, indicating that T_7 are neither redundant nor connected with M_a, therefore M_a = *(*T_7*.from,* T_7*.to)*.

While checking T_1, since T_1*.to>*M_a*.from*, indicating that T_1 is not connected with M_a, thus M_a is set to *(*T_1*.from,* T_1*.to)*.

Onto the checking of T_6, M_a*.from* = T_6*.to*, indicating T_6 is connected with M_6, therefore T_6 is merged into M_a, thus M_a*=(*T_6*.from,* M_a*.to)*.

T_5*.to* < M_a*.from*, indicating that T_5 is not connected with M_a, thus M_a = *(T5.from, T5.to)*.

Onto the checking of T_9, M_a*.from=*T_9*.to*, thus M_a = *(*T_9*.from,* M_a*.to)*.

Onto the checking of T_4, it is found that both T_4*.from* and T_4*.to* equals to those of M_a, thus T_4 is considered redundant, and is removed.

Finally, T_3 passed the checking with M_a, thus the redundant check is completed. The final result is shown in Fig. 5.

Note that the goal of this approach focuses on rapidly establishment of non-conflict tunnels. Once a tunnel is established, its attributes could be determined right away.

Since this algorithm shares the same goal with both Order-Split and Conflict-Free approach, and since these two are proven so far to be out-perform other approaches, thus here Zero-Conflict is compared with them. Shown in Table 4 are the numbers of resulting tunnels of Order-Split, Conflict-Free, as well as Zero-Conflict, generated from requirements in Table 2. It can be observed that Zero-Conflict yields same results for this case.

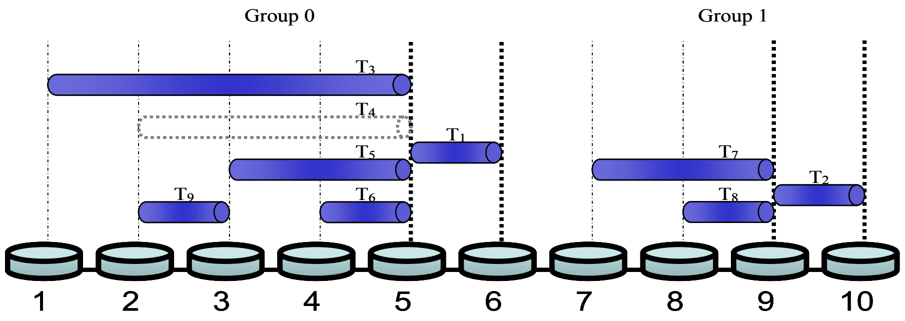


Fig. 5. The Solution for the Example of **Table 2** by Using Zero-Conflict Algorithm

Table 4. The Compare of Three Algorithms

Approach	Total Number of Tunnels
Ordered-Split Approach	8
ï ï ï ïïïï ïïï ïï ï ïAlgorithm	8
Zero-Conflict Algorithm	8

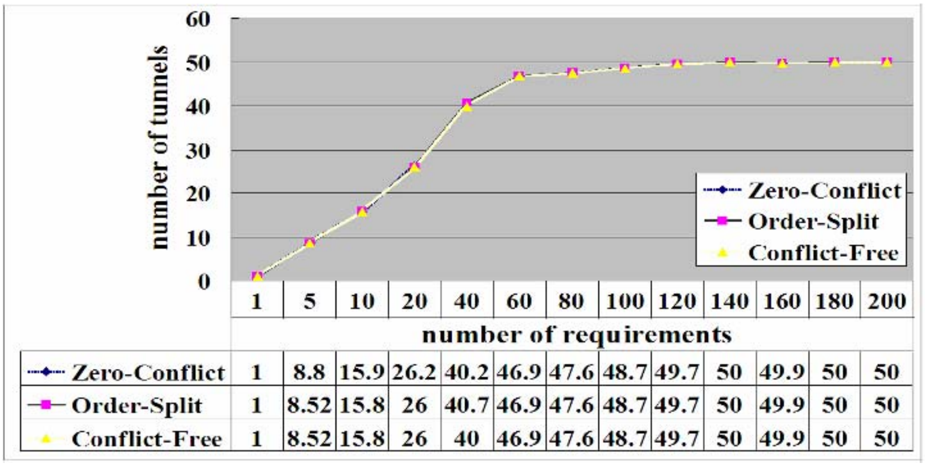
5 Time Complexity Analysis

The proposed Zero-Conflict Algorithm generates cut points right after security requirements are acquired. Checking for conflicts are unnecessary, since possible cases are successfully avoided. Removing of 1-hop requirements, grouping, and the three phases of tunnel building, are all *O(n)* operations. However several steps in the algorithm employed sorting operation, such as the sorting of requirement list, and gathering of the cut point list, which raised the over-all time-complexity to *O(nlogn)*.

In the final redundancy removal, the generated tunnels are sorted. It should be noted that in this approach, *n* input requirements will generate at most *2n* tunnels. Assuming there are *x* 1-hop requirements in these *n* requirements, then there will be at most *n-x* bus tunnels, *n-x* branch tunnels, and *x* 1-hop tunnels. Therefore the maximal number of generated tunnels is *2n-x*. Since $x \leqq n$, it is thus proven that *n* requirements generated at most *2n* tunnels, making redundancy removing itself a *O(nlogn)* operation. Thus the time-complexity of Zero-Conflict is bounded in *O(nlogn)*, which, in comparison with Order-Split and Conflict-Free, is significantly more efficient, as well as scalable.

6 Simulation Results

To show that Zero-Conflict, in addition of being fast, generates no more tunnels then existing approaches, a simulation was conducted. The simulator for Zero-Conflict algorithm was implemented under Windows platform. The simulation program takes a requirement file as input, and outputs a file containing generated tunnels. The Order-Split and the Conflict-Free approaches were also implemented. The performances of these algorithms were tested with inputs ranging from 1-200 requirements, each randomly generated 1000 times. The results of average amount of tunnels are shown in Fig. 6. The X-axis represents the number of the requirements input, while the Y-axis represents the number of tunnels generated. It could be seen that the result of Zero-Conflict is close Order-Split and Conflict-Free. Noted that under the assumption that the end nodes of all requirements are cut points, a tunnel covering *n* cut points will be divided into *n+1* connecting tunnels, which would raise the number of resulting tunnels. However, the simulation results show that the average number of tunnels generated by the proposed Zero-Conflict approach meets (or in some cases, beats) the results of most known approaches.



	1	5	10	20	40	60	80	100	120	140	160	180	200
◆ Zero-Conflict	1	8.8	15.9	26.2	40.2	46.9	47.6	48.7	49.7	50	49.9	50	50
■ Order-Split	1	8.52	15.8	26	40.7	46.9	47.6	48.7	49.7	50	49.9	50	50
▲ Conflict-Free	1	8.52	15.8	26	40	46.9	47.6	48.7	49.7	50	49.9	50	50

Fig. 6. The Average Number of Tunnels in the Network of 50 Routers

7 Conclusion

This paper proposed a Zero-Conflict algorithm, an automatic policy construction algorithm which is able to predict and avoid conflict in advance by using *requirement groups* and *cut points* techniques. Moreover, the worse case of time-complexity of this approach is only *O(nlogn)*, which as far as we know, beats most known approaches, whose worse cases of time-complexity are at least $O(n^2)$. Thus it is shown that by avoiding possible cases for conflicts, this approach yields both satisfying efficiency as well as effectiveness so that the resource for network management and the performance of the entire network is further improved.

In addition, most preceding algorithms are suitable for *central* processing, whereas security requirements are dealt with only after *all* of them are collected. The proposed concept of cut point prediction is more suitable for distributed processing. Future works can be made on utilizing this concept on constructing distributed processing algorithms.

References

1. M. Blaze, A. Keromytis, M. Richardson, and L. Sanchez, "IP Security Policy (IPSP) Requirements," RFC 3586, IPSP Working Group, August 2003.
2. C. L. Chang, Y. P. Chiu, and C. L. Lei, "Automatic Generation of Conflict-Free IPSec Policies," International Conference on Formal Techniques for Networked and Distributed Systems, pp. 233-246, October 2005.
3. J. Conover, "Policy-Based Network Management," Network Computing, Vol. 10, No. 24, pp. 44-50, November 1999.
4. Z. Fu and S. F. Wu, "Automatic Generation of IPSec/VPN Security Policies in an Intra-Domain Environment," 12th International Workshop on Distributed Systems: Operations & Management (DSOM 2001), pp. 279-290, 2001.
5. Z. Fu, S. F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine, and C. Xu, "IPSec/VPN Security Policy: Correctness, Conflict Detection, and Resolution," IEEE Policy 2001 Workshop, pp, 39-56, 2001.
6. H. Hamed, E. Al-Shaer, and W. Marrero, "Modeling and verification of IPSec and VPN security policies," 13th IEEE International Conference on Network Protocols (ICNP 2005), Vol. 0, pp. 259-278, November 2005.
7. S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," RFC 2401, Internet Society, Network Working Group, November 1998.
8. M. Li, "Policy-based IPSec management, " Network, IEEE, Vol. 17, No. 6, pp. 36-43, November 2003.
9. J. D. Moffett, "Requirements and Policies," Position paper for Workshop on Policies in Distributed Systems, HP- Laboratories, November 1999.
10. J. D. Moffett and M. S. Sloman, "Policy Hierarchies for Distributed Systems Management," IEEE Journal on Selected Areas in Communication, Vol. 11, No. 9, pp. 1404-1414, December 1993.
11. M. Sloman, "Policy Driven Management for Distributed Systems," Journal of Network and Systems Management, Vol. 2, No. 4, pp. 333-360, December 1994.
12. Y. Yang, C. U. Martel, and S. F. Wu, "On Building the Minimal Number of Tunnels - An Ordered-Split approach to manage IPSec/VPN policies," 9th IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Vol.1, pp. 277-290, April 2004.
13. Y. Yang, Z. Fu, and S. F. Wu, "BANDS: An Inter-Domain Internet Security Policy Management System for IPSec/VPN," 8th IFIP/IEEE International Symposium on Integrated Network Management 2003, pp. 231- 244, March 2003.

Conflict Prevention Via Model-Driven Policy Refinement

Steven Davy[1], Brendan Jennings[1], and John Strassner[2]

[1] Telecommunication Software & Systems Group,
Waterford Institute of Technology, Cork Road, Waterford, Ireland
{sdavy, bjennings}@tssg.org
[2] Motorola Labs, Chicago, IL, USA
john.strassner@motorola.com

Abstract. This paper describes an approach for application specific conflict prevention based on model-driven refinement of policies prior to deployment. Central to the approach is an algorithm for the retrieval of application-specific data from an information model relating to the subject and targets of a given policy. This algorithm facilitates the linkage of policies loosely defined at a high level of abstraction to detailed behavioural constraints specified in the information model. Based on these constraints policies are then modified so that conflicts with other deployed policies can be readily identified using standard policy conflict detection techniques. This approach enables policy enforcement to be cognisant of application specific constraints, thereby resulting in a more trustworthy and dependable policy based management system.

1 Introduction

This paper presents an approach for refinement of newly created or modified policies so that application specific conflicts with already deployed policies can be readily prevented. We propose the use of a policy analyser that can interrogate an information model containing detailed information about the system for which policy is being defined, and use this information to refine the high level policy into a policy embodying information regarding system constraints its actions may be subject to.

This paper describes the operation of a policy analyser, and a prototype implementation demonstrating its use in a policy based management system. The paper is structured as follows. §2 discusses current work on policy conflict detection and prevention, and on methods for analysing information contained within an information model. §3 presents an architecture for policy conflict prevention and specifies the algorithms for retrieval of relevant information and policy refinement. Our prototype implementation is described in §4, whilst its operation in an experimental test bed is described in §5. Finally, §6 summaries the paper and outlines topics for future work.

2 Related Work

This section discusses published work in the domains of policy conflict analysis and information model processing.

2.1 Policy Conflict Analysis

Policy conflict detection and resolution is a necessary component of any Policy Based Management System (PBMS). A PBMS must employ a facility to verify that newly created or modified policies conform to intended system behaviour before they can be deployed. From the perspective of our approach, a policy conflict can be seen to be a potential occurrence of unintended behaviour within the PBMS. This can manifest itself in many forms. Most have been documented by Charalambides, et al. [1], who categorise conflicts as domain independent or application specific. Domain independent conflict analysis can be carried out by offline processes that indicate whether conflicts will definitely occur or may occur in a specific context. If we can detect the conditions in which a conflict can occur, then we can resolve the conflict by either modifying or removing one or more conflicting policies. The issue, of course, is if we have enough knowledge to detect all conditions in which a conflict can occur. For example, if conflicts are known at design time, then one can devise strategies to deal with them. However, in networking, one often encounters conflicts at run-time which were not envisaged during the design period. Hence, the challenge is to design a robust conflict detection approach that can deal with unforeseen situations.

Detection of application specific conflicts requires more information about the system for which policies are being defined. In [2] the authors augment the PBMS with extra information, expressed as rules relating to the managed entities. These rules are triggered when an application specific conflict is about to occur; such conflicts are resolved based on specific resolution policies associated with each of these rules. This approach depends on the policy author both being able to specify system constraints that policies must adhere to, and being able to translate these constraints into the appropriate custom rule format. In [13] Shankar and Campbell use pre-conditions and post conditions to describe the effects specific actions will have on a system, they use this axiomatised rule-actions to help in the conflict prediction process. These again have to be encoded into the policies to be effective.

2.2 Information Model Processing

An Information Model is a representation of managed entities, concepts and their relationships independent of platform, language, and protocol. Information models play a central role in network management and considerable efforts have been expended on the specification of standard information models. One of the more mature standards is the TM-Forum's Shared Information and Data Model, which is closely related to DEN-ng [3]. One of the main advantages of DEN-ng is its extensive use of patterns and abstractions (such as roles) to allow behaviour to be defined and *orchestrated* over the associated components of the system being described. Use of an information model of a system to aid in policy based management is also described in [4], aimed at managing specifically IP networks, and more recently towards autonomic communication networks [5].

For model-driven policy refinement we are specifically interested in efficient retrieval of relevant information from a system model. For UML-based models like SID/DEN-ng a number of approaches for information retrieval exist. One such method described in [11] details how the UML artefacts used to build a class diagram describing an information model can be translated to an ontology where it is represented in OWL (Web Ontology Language). This ontology can then be reasoned over and queried using semantic web technologies. A benefit of this approach is the ability to use existing ontologies to expand the information model such as linking to user profiles.

Another approach would be to translate the UML into an XML format such as XMI (XML Metadata Interchange) [8] where it can be efficiently queried over using XQuery. XQuery provides an efficient method of querying repositories of XML documents within an XML database. Meier [9] describes the performance of such an XML database called eXist, where test documents of about 40 Megabytes can be efficiently queried. Information model repositories generated from UML to XMI are not expected to reach this size.

3 Description of Approach

Model-driven policy conflict prevention is the process of refining newly created or modified policies so that conflicts with already deployed policies can be readily detected using standard policy conflict detection approaches. Policy refinement in this context involves the specification of additional condition clauses within the policy, which subsequently allows the detection of conflicts with other policies that would otherwise have gone undetected by standard policy conflict detection algorithms.

More specifically, in cases where system information models describe constraints relating to the operation of managed entities, relevant policies can be augmented with conditions reflecting these constraints, so that they will not be enforced in a manner that results in these constraints being violated. System constraints in the information model are defined by the system architect who has expert knowledge in the functionality of the system being modelled. These system constraints may come in the form of action pre-conditions, invariants, or post-conditions. However the policy authors, be they business analysts or network administrators, have vastly differently views of the system for which they are defining policy. Therefore they have an incomplete view of the system as a whole. System constraints defined within the information model can help bridge this gap by supplying implicit knowledge not usually available to the policy authoring process. Our approach is to introduce an automated policy refinement process which obviates the need for policy authors to be cognisant of the detailed constraints on system operation, but which outputs policies that are sufficiently well specified that policy conflict detection processes can be effective and efficient. Our approach is primarily concerned with action pre-conditions or action constraints.

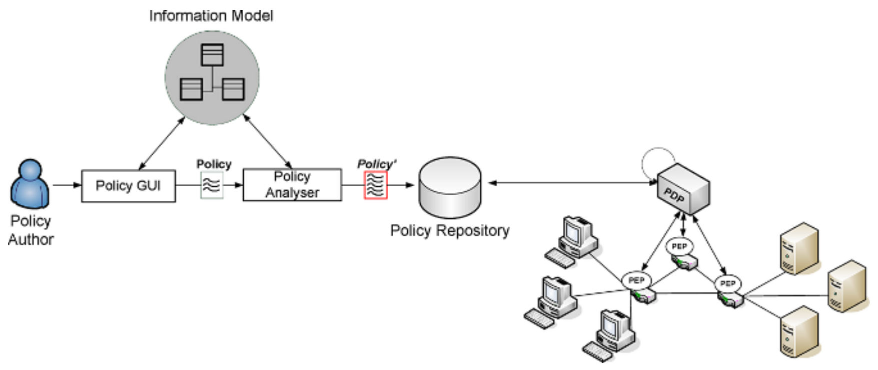


Fig. 1. PBMS Architecture incorporating model-driven conflict prevention

3.1 PBMS Architecture Incorporating Policy Conflict Prevention

Fig. 1. illustrates a PBMS architecture incorporating model-driven conflict prevention. We now briefly describe the role of the Policy GUI, the information model and the Policy Analyser. The Policy GUI is the interface used by policy authors who are primarily concerned with ensuring that services and resources are managed in a manner consistent with business objectives and goals. Policy authors are likely to be business analysts who define or modify policies relating to particular customers and their access to the services provided by the network. They are unlikely to have the detailed knowledge of the network required to specify policies at the level of detail required for easy detection of conflicts with other deployed policies.

The information model describes, in a platform independent manner, the characteristics and behaviour of the different managed entities comprising the managed environment, as a set of related *model elements*. Model elements include classes, attributes, relationships, constraints, and other artefacts. For example, the information model will describe which customers can use which services where and how. Constraints within the information model can be described using a constraint language like the Object Constraint Language (OCL) [7]. OCL specifies constraints using invariants, pre-conditions and post conditions associated with all attributes, associations and operations on each modelled class.

Policies created or modified by policy authors are expressed in strict accordance with the terms used in the information model, since the policy GUI is tightly coupled to the information model, as described in [5]. Once created/modified policies are passed to the Policy Analyser, which takes their subjects and/or targets and queries the information model for relationships (and constraints on these relationships) for these subjects/targets. Using relationship and constraint information it is possible to assess more precisely those circumstances in which the policy actions should be invoked. To achieve this, the Policy Analyser employs an algorithm that retrieves the relevant relationships and constraints from the information model given an arbitrary

```
Inputs [Policy]
Outputs [Relationships and Constraints]
   List Subjects defined in Policy
   List Targets defined in Policy
   List Actions defined in Policy
   For every element of Subjects
          Subject Managed Entities = Look up the corresponding Class
          descriptions from the Information Model
   For every element of Targets
          Target Managed Entities = Look up the corresponding Class
          descriptions from the Information Model
   For every element of Target Managed Elements,
          If there is an Action requested by the Subject Managed Entity
          define within the Target Managed Entity that matches the
          Action in the Policy then add the pre-conditions of this
          action to the relationships and constraints list.
Return (Relationships and Constraints)
```

Fig. 2. Policy Action Constraint Retrieval Algorithm

policy defined in accordance with that information model. Such an algorithm is specified in §3.2 below.

3.2 Policy Action Constraint Retrieval Algorithm

In specifying an algorithm for policy action constraint retrieval we firstly assume that policies specify the policy subject using the terms used in the information model (e.g. there must be a one-to-one, or one-to-many, mapping between a policy subject and a class in a UML based model). The target(s) of the policy, if included, must also be similarly specified. If the target is not specified explicitly, it must be possible to infer it from the information model by examining the relationships between the subject and the actions. Finally, policy actions must map to relationships between those model artefacts representing the policy subjects/targets.

Given these assumptions the algorithm outlined in Fig.2 provides a means of discovering the relevant policy action constraints based on model artefacts and their relationships.

3.3 Policy Refinement Algorithm

Once the associated relationships and constraints have been retrieved, the original policy needs to be refined. As there may be multiple action constraints to be added into the policy, they must first be checked against each other so that the resulting policy action constraints do not logically contradict. An example of this would be if two constraints were added to a policy specifying that the action may only be performed during daytime hours, and another constraint specifying that the action may only be performed during night time hours. This type of rule contradiction will cause the policy not be enforced at anytime, and so the policy can not be refined and is invalid against the referenced information model. The constraints must also be checked against existing policy conditions for completeness.

```
Inputs [(Relationships and Constraints); Policy Conditions]
Outputs [newPolicy]
   For every Relationship select PolicyAction.PreConditionsConstraint
        A Pre Condition Constraint is selected from each Relationship
        and tested against all previously selected Constraints
        For each Constraint in Constraints and Policy Conditions
             If newConstraint AND Constraint
                is a Logical Contradiction
             Then
                The Condition Clause of the Policy will never be
                satisfied and the algorithm is aborted
             Else
                Add newConstraint to the list of Constraints
Combine the resulting list of constraints to the Policy Conditions as
new conditions
Return newPolicy
```

Fig. 3. Policy Refinement Algorithm

4 Prototype Implementation

The prototype implementation, depicted in Fig.4., will now be described. The Policy GUI is developed in Java, and enables the policy author to create high level policy using context sensitive drop down menus. A detailed description of this GUI can be found in [5]. The options available to the policy author are limited to the entities describe in the information model, so that subject, targets and actions must be specified in the information model before they can be used to define policies. The policies output from the GUI are defined from the view the policy author has of the managed system. This allows the policy author to only be concerned with authoring policy appropriate to his level of knowledge, while enabling the policy analyser to develop more specific instances of this policy.

The Policy Analyser is a Java process that is invoked on every new or modified policy. Access to the information model is performed by processing a set of XML files that represent the information model. The information model is initially described using a UML class diagram editor, and is exported to an XMI [8] format. XMI is the OMGs (Object Management Group) standard format for describing UML diagrams, however only the class diagram aspects of the standard are of interest for the moment. The information model constraints are defined in a separate OCL file. The OCL constraints are translated from managed entities action pre-conditions into policy conditions that can be understood by the policy repository and policy analyser via Kent OCL library [10]. This library provides java class implementation of OCL constraints that can be evaluated in real-time. The policy repository takes two forms; policies are stored in an XML format for query and retrieval using eXist XML database for storage and XQuery for searching; they are also stored in a JBoss rules engine in working memory, where reasoning over policies is performed. Policies stored in the JBoss rules engine [12] are encoded as Java Bean objects, so a simple policy class hierarchy is used. The JBoss rules engine also holds a runtime

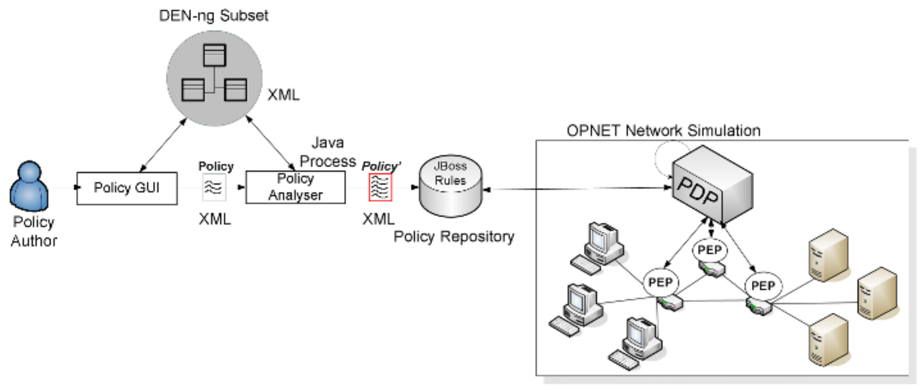


Fig. 4. Prototype Implementation

representation of the data defined in the information model, such at router information and link information which is updated at regular intervals.

Some simple policy types are defined such as permit, obligation, and refrain. Policies added to the JBoss rule engine can be rapidly reasoned over to discover whether there are any domain independent conflicts, such as a conflict of modality. The rule engine can also detect if two policies referring to the same action and target will potentially cause a conflict when the conditions are satisfied.

The system being managed is simulated with OPNET, allowing for flexibility at the network level where it is easy to modify the underlying network scenario. PDPs receive updated policy and enforce it through the simulated PEPs. A more detailed description of the simulated system is provided in [5].

5 Scenario and Results

This section describes a scenario where there are two customer networks subscribed to services provided by a single Internet Service Provider (ISP). Our ISP has defined a simple information model (using a subset of DEN-ng) and policies as follows.

5.1 High Level Policies and Information Model

The policies will describe the conditions as to when a certain customer is permitted to request provision of RTP (Real Time Protocol) traffic for its usage.

There may be several similar policies defined for other customers of the system where they too are permitted to request the allocation of bandwidth. There may also be policies defined not by the business user but by the network administrator that will also require the allocation of bandwidth. When the defined policy in Fig.5 is enforced, the core network will modify the PHB (per hop behaviour) of the edge and core routers to reflect the provision of the requested service. We can see this interaction modelled in the information model in Fig. 6 below. As Fig. 6 shows, a customer can

```
<policy name="WITServicePolicy" type="permit">
     <subject type="Customer">WIT</subject>
     <event type="From">09:00</event>
     <event type="To">17:00</event>
     <event type="Trigger">RequestRTPSession</event>
     <operation>
             <target type="RouterLink"/>
             <action type="AllocateBW">
                     <param name="grade" value="1"/>
                     <param name="amount" value ="5Mbps"/>
             </action>
     </operation>
     <condition/>
</policy>
<policy name="TSSGServicePolicy" type="permit">
     <subject type="Customer">TSSG</subject>
     <event type="From">08:00</event>
     <event type="To">16:00</event>
     <event type="Trigger">RequestRTPSession</event>
     <operation>
             <target type="RouterLink"/>
             <action type="AllocateBW">
                     <param name="grade" value="1"/>
                     <param name="amount" value ="4Mbps"/>
             </action>
     </operation>
     <condition/>
 </policy>
```

Fig. 5. High Level Policies

subscribe to the RTP service which uses resources such as the EdgeRouter and the CoreRouter.

Focussing on the RouterLink managed entity; there are two operations available for this scenario – allocation and deallocation of bandwidth. We will now discuss the former, as the latter is very similar. `AllocateBW()` will instruct the nested core and edge routers to configure their PHBs to reflect the request. As there are always limited resources on the network, we cannot keep calling `AllocateBW()` and expect bandwidth to be always available to allocate. OCL is used to define the semantics of these attributes and the following OCL is attached to the `AllocateBW()` operation to constrain its use concerning how bandwidth can be allocated.

```
context RouterLink::AllocateBW(ToS:Integer, amount:Real)
pre perserveBWLimit: self.currentBW + amount <self.maxBW
```

When the original policy is run through the policy analyser it is refined with information describing more accurately when the policy should be actually enforced. The algorithm defined in Fig.2 is implemented as a set of XQuery functions where the policy document is input. For example the subjects of the policy can be discovered using the XQuery terminology, `doc("policy.xml")/policy/subject/@type`, which will return a type represented by the subject mention in the policy. Similar statements can retrieve the targets and actions of the policy. XQuery is also used to query the

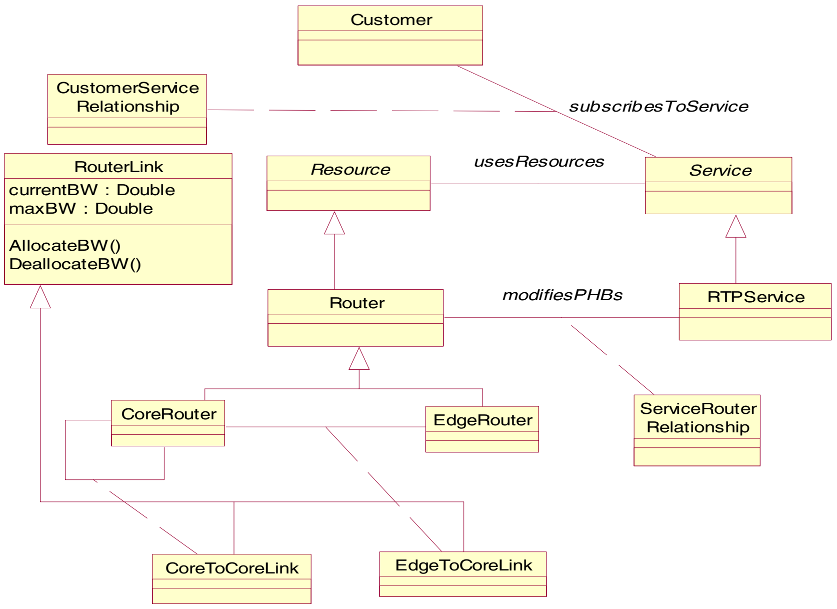


Fig. 6. Den-ng Subset Information Model

XMI representing the information model. To look up a class entity's id the query below can be used.

```
for $x in doc("InfoModel.xmi")//*[@name]
     where (compare(name($x),'UML:Class') = 0) and (compare($x/@name,
     'Service') = 0)
     return string($x/@xmi.id)
```

Once we have the id of the policy entities we can then discover further associations, and relationships between other entities using the information model. The algorithm finishes with selecting the appropriate OCL from the OCL files; this is easily carried out because every OCL statement includes a context mentioning the

```
<policy name="WITServicePolicy" type="permit">
     <subject type="Customer">WIT</subject>
     <event type="From">09:00</event>
     <event type="To">17:00</event>
     <event type="Trigger">RequestRTPSession</event>
     <operation>
            <target type="RouterLink"/>
            <action type="AllocateBW">
                   <param name="grade" value="1"/>
                   <param name="amount" value ="5Mbps"/>
            </action>
     </operation>
<condition>RouterLink.currentBW + 5 < RouterLink.maxBW</condition>
</policy>
```

Fig. 7. Modified Policy

reference class and actions it is constraining. The Kent OCL library then processes the OCL and generates the extra policy conditions required to refine the associated policy. The policy in Fig.7. is a refined policy from Fig.5.

The policy defined in Fig.7 describes an extra condition of which the original policy author would not be aware. The clause is evaluated in real-time when the policies are being processed to see if they apply at the current situation. This new information will further constrain when the policy will be valid. The Kent OCL library generates a java bean that will evaluate this condition for the JBoss rule engine during analysis and at runtime.

5.2 Policy Enforcement

Suppose that the two original policies were deployed to the system, and currently the `currentBW` and `maxBW` of the related RouterLinks are 0.0Mbps and 8.0Mbps respectively. An event of type Request RTP is initiated by the customer WIT at approximately 08:15am. This event triggers the enforcement of the relevant policies allocating 5Mbps of bandwidth over the related RouterLinks (first policy enforcement in Fig. 8.). An event of type Request RTP is then initiated by the customer TSSG at approximately 9:40am (second policy enforcement in Fig. 8.). This triggers an attempt to allocate a further 4Mbps of bandwidth on the related links. However an application specific conflict occurs that was not detected previously, whereby more bandwidth is being allocated than is available. The effects of allowing this conflict to go "untreated" are unpredictable, as the situation is not catered for. From Fig.8 we see that the allowable capacity of the core link is 8 Mbps, and as the new RTP session was allowed, it can only be partially met. Also, this will adversely affect other existing sessions.

Now suppose the original policies were analysed and refined to reflect the constraints specified within the information model. The policy information added in

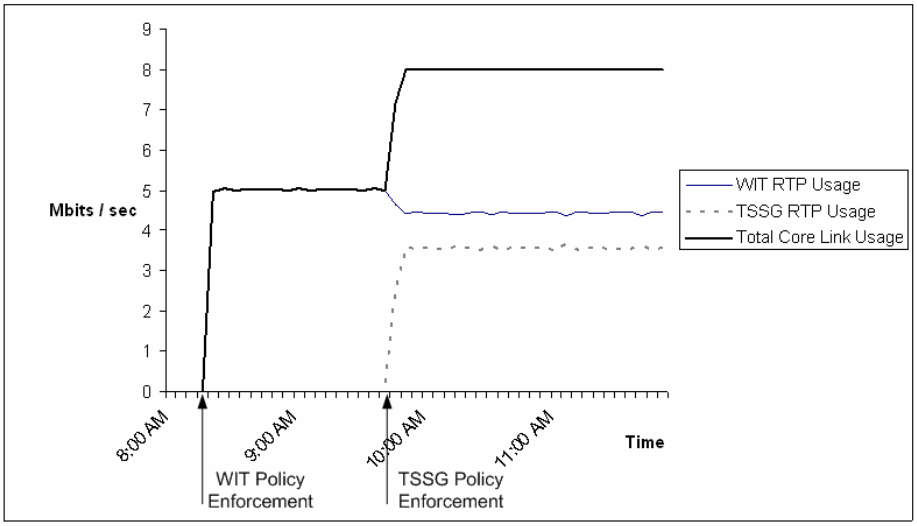


Fig. 8. Application Conflict Illustration

Fig.7 is added to both policies. In this updated scenario, the first event still succeeds, but the second event does not trigger the permit policy and is discarded, as the policy will not meet all of its conditions. Specifically, when the condition clause of the policy is checked, it is evaluated to false because the `currentBW` plus the requested bandwidth exceeds the `maxBW` of the related RouterLinks.

6 Conclusions and Future Work

Policy conflict situations, when not catered for, will allow the system being managed to produce unpredictable behaviour. This is an undesirable scenario for potential ISPs looking to employ policy based management to control the behaviour of their network. This paper introduces an architecture and prototype implementation that refines high level business policies with application specific information so that conflicts can be readily detected. This form of conflict prevention is made possible using an information model defined over the services and resources of the system, where the constraints of the system are defined by a domain expert. Algorithms that process a policy in order to retrieve constraint information and subsequently refine the policy are defined and implemented. A model-driven approach to refining policies towards conflict prevention frees the business user from being concerned with the behavioural details of the core network, and introduces a level of safety and dependability into the system. One potential downside is that certain business policies may not be enforced as originally described, thus provision of appropriate feedback to the policy author would be desirable.

Future work will be focused on developing our algorithm to be used with existing policy languages and policy based management systems such as Ponder [6]. We also intend on developing a richer information model along with a set of obligation, permit and refrain policies to investigate what other information can be used from the information model to aid in conflict prevention. We also intend on exploring other aspects of Information Models that define system behaviour such as flow charts and finite state machines.

Acknowledgements

The authors would like to take this opportunity to thank the anonymous reviewers for their useful comments and feedback on the paper. This work has received support from the Science Foundation Ireland under the Autonomic Management of Communications Networks and Services programme (grant no. 04/IN3/I404C).

References

1. Charalambides, M. et al., “Policy Conflict Analysis for Quality of Service Management,” in Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks POLICY'05, Stockholm, Sweden (2005) 99-108
2. Charalambides, M. et al., “Dynamic Policy Analysis and Conflict Resolution for DiffServ Quality of Service Management” in Proceedings of the IEEE/IFIP Network Operations and Management Symposium 2006, Vancouver, Canada (2006) 294-304

3. Strassner, J., “Policy-based Network Management: Solutions for the Next Generation”, Morgan-Kaufman Publishers. ISBN 1-55860-859-1 (2004)
4. Strassner, J., “Directory Enabled Networks”, Macmillan Technical Publishing, ISBN 1-57870-140-6, (1999)
5. van der Meer, S., Davy, A., Davy, S., Carroll, R., Jennings, B., Strassner, S. 2006, "Automonic Networking: Prototype Implementation of the Policy Continuum", in Proc. Workshop in Broadband Converged Networks at IEEE/IFIP Network Operations & Management Symposium, Vancouver, Canada (2006)
6. Damianou, N., Dulay, N., Lupu, E.C., Sloman, M., “The Ponder Specification Language,” presented at 2nd IEEE Workshop on Policies for Networks and Distributed Systems, Bristol, UK (2001)
7. OMG, UML 2.0 OCL Specification v2.0, Object Management Group Specification, Available at http://www.omg.org/docs/formal/06-05-01.pdf: accessed July (2006)
8. OMG, Meta Object Facility (MOF) 2.0 XMI Mapping Specification, v2.1, Object Management Group Specification, Available at http://www.omg.org/docs/formal/05-09-01.pdf accessed July (2006)
9. Meier, W., eXist: An Open Source Native XML Database, In Web, Web-Services, and Database Systems. NODe 2002 Web- and Database-Related Workshops, Erfurt, Germany (2003) 169-183
10. Akehurst, D., Patrascoiu, O., OCL 2.0 – Implementation the Standard for Multiple Metamodels”, Workshop Proceedings, 6th International Conference on the Unified Modeling Language and its Applications, UML2003, Electronic Notes in Theoretical Computer Science (2003)
11. Lehtihet, E., Strassner, J., Agoulmine, N., O Foghlu, M., Ontology-Based Knowledge Representation for Self-Governing Systems, accepted for publication in 17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2006, Dublin, Ireland, October (2006)
12. JBoss Rules, JBoss, Available at http://labs.jboss.com/portal/jbossrules/ accessed August (2006)
13. Shankar, C., and Campbell, R., A Policy Based Management Framework for Pervasive Systems using Axiomatized Rule-Actions in Proceedings of the 2005 Fourth International Symposium on Network Computing and Application (NCA’05), Cambridge, Massachusetts, July (2005) 255-258

Minimum-Intrusion Approaches for In-Service BER Estimation in Transparent WDM Networks

Carolina Pinart

Centre Tecnològic de Telecomunicacions de Catalunya
carolina.pinart@cttc.cat

Abstract. The bit error rate (BER) is a key measure to quantify the reliability of a transmission system. In transparent networks, link bit error checks are not cost-effective. This paper proposes approaches for low-cost, real-time BER estimation with minimum opto-electronic conversions and applies them to a transparent optical networking testbed[1].

1 Introduction

Today's optical transport networks are based on the Synchronized Digital Hierarchy (SDH) [1]. Electrical regeneration may amount to 70-90% of the cost of lighting up a new wavelength in an SDH network [2]. Therefore, the removal of opto-electrical (O/E) conversions in core nodes, which is known as transparency, will result in the efficient transportation of any type of data traffic (predominantly based on the Internet Protocol, IP), regardless of its payload or format. Moreover, equipment for Wavelength Division Multiplexing (WDM), tunable lasers, reconfigurable optical cross-connects and optical add-drop multiplexers, along with emerging approaches of optical intelligence, have matured sufficiently to build ultra-high-capacity networks. Future optical networks are also expected to provide new on-demand connectivity services with different quality levels. In the context of transparency (analog transmission), this results in a major challenge for in-service performance monitoring due to the lack of electrical regeneration in the core elements, which limits the amount of monitoring information available, especially a paramount digital parameter: the Bit Error Rate (BER).

In the first deployment phase of transparent networks, each WDM connection is expected to transport a single service, which is known as a wavelength-based or lambda service. A WDM connection is an amplified intensity-modulation, direct-detection (IM/DD) system [3][4]. Physical-layer (layer 1, L1) quality of service (QoS) will be crucial here in the sense that each service class will have to be defined by a set of parameters characterizing the quality of the optical signal transporting it; a wavelength-based Service Level Agreement (SLA). Defined as the ratio of errored bits to the number of transmitted bits, the BER captures the overall performance of the physical layer, and is a basic SLA parameter.

[1] This work was partially funded by the Spanish Ministry of Science and Education through the project RESPLANDOR under contract TEC2006-12910/TCM.

However, in practice it takes long to calculate the BER in terms of received bits. Therefore, the BER is usually estimated in real time by performing bit and block checks at each link, that is, in edge and core SDH nodes. In a transparent network, such checks are only possible at the edges, which makes on-line link BER estimation a challenge. Service providers are beginning to use non-intrusive monitoring (NIM) capabilities in their WDM networks to determine physical-layer performance metrics that measure the integrity of optical signals without electrical regeneration. Among them, the Optical Signal to Noise Ratio (OSNR) seems a good candidate to estimate the BER [3] [5]. Network-layer parameters, such as the Packet Error Rate (PER), are also candidates for in-service monitoring.

This paper proposes scenarios with minimum or absence of O/E conversions to estimate the BER in transparent networks using non-intrusive capabilities where possible and provides a practical example in the form of a laboratory testbed. The remainder of the paper is organized as follows. In Section 2 we propose three monitoring scenarios for detecting bit errors at layers 1, 2 and 3 in transparent WDM networks. Section 3 provides a practical example of BER estimation according to the scenarios of Section 2 that do not require O/E at the core nodes, which is implemented in the ADRENALINE testbed, a transparent network that supports QoS-enabled services monitored non-intrusively.

2 Scenarios for BER Estimation in Transparent IM/DD

When offering a transparent lambda service, traffic is mapped natively onto individual wavelengths. The service is logically and physically terminated directly onto the end user's IP router or layer 2/3 (L2/L3) switch, and is transported across an individual wavelength over the transparent network to be terminated on another IP router or L2/L3 switch. This transparency allows the delivery of the service to be more cost-effective but at the same time it makes the measurements of SLA metrics difficult to implement. Without optical-based monitoring capabilities, measurement of QoS in transparent networks is reduced to measuring the physical connectivity. With SLA metrics being provided on a per-service basis, service monitoring and SLA measurements will have to be implemented on each individual wavelength. Real-time performance monitoring and management will be essential in this context, which results in the following requirements:

1. Accurate monitoring of the raw bit stream in real time at multi-gigabit rates.
2. Independence of the bit rate.
3. Use with high number, dense-spaced, multiple-bitrate WDM channels.
4. Rapid detection of degradation and proactive response.
5. Limited latency and/or overhead.

Apart from these requirements, monitoring should not defeat transparency (i.e., be non-intrusive) and be low-cost. The rationale behind this is twofold: independence of bitrate and format, and low capital and operational expenses.

2.1 L1/L2 Monitoring (Electrical)

- **Framework:** A carrier owns one or more transparent networks (from source to destination ports of two IP routers).
- **Challenges:** This scenario resembles bit/block error measurements in the receiving ends of SDH networks [1]. For example, the Optical Transport Network [6] uses the digital wrapper (DW) to multiplex data streams from various sources into common telephony-based payloads. Multiple data streams from different sources are mapped into the same DW bandwidth as time domain multiplexing (TDM) payloads. The information in the communications stream is multiplexed at the physical layer (TDM payload).
- **Solutions:** If using SDH or Generic Frame Procedure (GFP), bit/block error measures [1]. If using Gigabit Ethernet (GigE), parity check. Another option is the estimation of BER from the received electrical signal [4].
- **Monitoring:** This scenario requires intrusive monitoring (O/E conversions). Some IP routers have embedded GigE/SDH/GFP framing capabilities. Alternatively, devices for bit/block error check and/or Signal to Noise Ratio (SNR) testers with embedded BER estimation must be employed.

2.2 L1 Monitoring (Optical)

- **Framework:** Same as previous scenario.
- **Challenges:** The main complication in this scenario is that the performance measurements available, which are typically limited to optical power, OSNR and wavelength registration, do not directly relate to QoS measures used in SLAs. Since the monitoring system only accesses the optical layer, no parity checks or SNR measurements are possible. Moreover, transparency means that it is not possible to access overhead bits in the transmitted data to obtain performance-related measures.
- **Solution:** Estimation of the BER from the OSNR. Since this solution is non-intrusive and performed in the optical domain, it can be applied at any point in the network by tapping a small portion of the transmitted WDM signal. The use of the channel OSNR ($ONSR_c$) as a means to estimate the BER of the signal (BER_c) is based on the assumption that the Q factor can be used as an intermediate parameter. Humblet and Azizoǧlu [3] derived widely-used approximate expressions for the Q factor as a function of the OSNR. While the Q factor can be directly converted to an electrical SNR value [4], the relationship to the OSNR is unfortunately not so simple. The study of Humblet and Azizoǧlu for ASK systems has the following result: $P_e = Q(\frac{2\frac{S}{N}}{\sqrt{4\frac{S}{N}+M}+\sqrt{M}})$, where *Q(x)* denotes the CDF of a zero mean, unit variance Gaussian random variable [3], and $2M = 2B_oT + 1$ and S/N is the SNR. Assuming M=1, and combining the results of [3] and Becker *et al.* [5], the relation between the Q factor and the OSNR can be approximated as:

$$Q = \sqrt{\frac{B_o}{B_e}} \frac{2OSNR_c}{\sqrt{4OSNR_c + 1} + 1} \tag{1}$$

where B_e is the electrical bandwidth of the receiver filter and B_o is the optical bandwidth. IM/DD systems with low inter-symbol interference and Gaussian noise distribution verify that the Q-factor expression [4] and eq. 1 are equal [3]. Gaussian distribution is used to model the ASE noise introduced by optical amplifiers as dominant over the receiver shot and thermal noises. Therefore, we obtain the channel BER (BER_c) from the channel OSNR [7].
– **Monitoring:** Optical Performance Monitoring (OPM) devices perform non-intrusive monitoring by tapping a WDM fiber. Commercial OPMs monitor several fiber ports, each supporting tens to hundreds of WDM channels. OPM monitors are integrated in edge and core nodes using optical splitters.

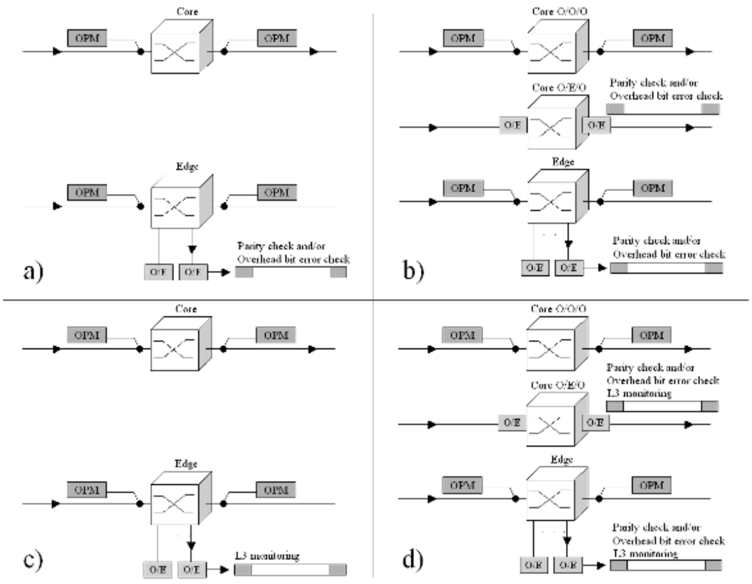


Fig. 1. Combination of scenarios for BER estimation in transparent networks

2.3 L3 Monitoring

– **Framework:** Customer-empowered fiber networks, which are becoming a reality due to the access to dark fiber resulting from the liberalization of leased line provisioning. Little effort on OPM is expected from these networks, which basically provide IP services (packet-level monitoring).
– **Challenges:** The Packet Error Rate (PER), defined as the rate at which errors in transmission/reception result in the rejection of a packet, is a standard measure of network-layer performance. Packet loss is the main SLA parameter monitored by users and service providers. This parameter can be monitored in real time with fairly good accuracy. In a transparent network, packet errors occur because of errors and impairments in the physical layer,

which cause data bits to toggle. A transparent WDM network is seen by the IP layer as a single hop, which means that network load, congestion avoidance mechanisms or IP header corruption do not cause packet losses, simply because they do not exist. Many research efforts have been put into reflecting the relationship between the raw PER and the link BER when the packet loss is a result of bits in error at the physical layer: $BER_c = 1 - \sqrt[s]{1 - PER}$, where s is the size (in bits) of a packet when no coding is done.

– **Solution:** The coding scheme affects the way in which bit errors on the physical layer propagate up the network stack. Both the errors occurring on a WDM connection and the protection scheme have an impact on the PER for the packets transmitted over that channel. In a low power regime, [8] shows that 8B/10B block-coding causes a non-deterministic relationship between PER and BER in optical GigE. Therefore, PER monitoring does not seem a substitute to BER monitoring, but rather a complement.
– **Monitoring:** For NIM, IP routers connected to edge nodes have embedded packet statistics capabilities. Otherwise, packet analyzers can be used. In core nodes, optical splitters (after demultiplexing) and packet analyzers are needed. For intrusive monitoring, IP test traffic generation and monitoring nodes are needed both in edge and core nodes.

In a network with N core nodes with F in/out fibers per node and C WDM channels per fiber, and M edge nodes with W channels per receiving end, the capital expenses for estimating the BER on-line ($c_{number\ of\ scenario}$) are:

– End-to-end (edge): $c_1 = Mc_{L1/L2}$; $c_2 = M(c_{OPM} + Fc_{splitter})$; $c_3 = Mc_{L3}$
– At each hop (core and edge nodes): $c_1 = 2NFCc_{O/E} + (N+M)c_{L1/L2}$; $c_2 = (N+M)c_{OPM} + (2N+M)Fc_{splitter}$; $c_3 = NFCc_{splitter} + (N+M)c_{L3}$

where $c_{O/E}$ is the cost of a transponder, $C_{L1/L2}$ is the cost of bit/block error count capability, $c_{splitter}$ is the cost of an optical splitter, c_{OPM} is the cost of an OPM monitor and c_{IP} is the cost of a dedicated device for packet statistics. For simplicity, in scenario 2 we assume that each optical node is equipped with a single multi-fiber OPM monitor. For scenario 3, we assume that L3 monitoring is done non-intrusively with a single packet-capturing device per receiving end and that no coding is done. Note that the O/E cost for the MW channels added/droped at the edge nodes is not included because it is an expense necessary for the operation of the network. Note also that scenario 1 is the only one that requires overhead to compute bit/block errors (e.g., GFP). Figure 1 illustrates possible combinations of the above scenarios with minimum O/E.

3 The Example of the ADRENALINE Testbed

The ADRENALINE testbed is a transparent dense WDM network developed at the Centre Tecnològic de Telecomunicacions de Catalunya. Each node is enabled with an OPM monitor that measures channel power, frequency drift and OSNR of the in/out fibers. A broadband tester measures L3 statistics for IP traffic.

3.1 On-Line SLA Validation in ADRENALINE

ADRENALINE supports three service types, whose QoS of these services can be verified in real-time by a NIM system [7]. The monitoring scenario of the ADRENALINE testbed is novel because it combines pure non-intrusive scenarios at L1 and L3 (scenarios 2 and 3, Figure 1c). The rationale is to accomplish the monitoring goals listed in Section 2 and to build a solution that supports easy migration to full OPM once the current technological limitations are removed:

1. The OSNR is obtained by NIM and allows link BER estimation.
2. Spectral monitoring is bitrate-independent.
3. Non-intrusive OPM allows monitoring of DWDM channels in milliseconds.
4. Non-intrusive OPM allows detecting degradations such as OSNR levels and power losses in milliseconds. Suitable fault location algorithms are needed for proactive response due to the propagation of faults in transparent networks.
5. L1 and L3 NIM add neither overhead nor latency.

Moreover, L3 NIM at the edge nodes results in a minimum amount of O/Es and low cost by using embedded packet statistics capabilities of the routers. The cost of on-line BER estimation in the testbed is $c_{ADRENALINE} = 3c_{OPM} + 12c_{splitter}$. Note that no overhead is added to the transported data for monitoring purposes, because no bit error checks are done. On the other hand, this model relies on the OSNR as the means to estimate the BER. In some cases, the difference between the real and estimated BER may be too large. Then, it may be interesting to add an offset or to compensate estimation errors.

References

1. "Network Node Interface for the Synchronous Digital Hierarchy (SDH)," ITU-T Rec. G.707, 1996.
2. S. Barnes, "All-optical networks: principles, solutions and challenges," in *in Proc. Optical Fiber Communication Conference (OFC)*, 2002.
3. P. A. Humblet and M. Azizoglu, "On the bit error rate of lightwave systems with optical amplifiers," *IEEE/OSA Journal of Lightwave Technology*, vol. 9, no. 11, pp. 1576–1582, November 1991.
4. D. Marcuse, "Derivation of analytical expressions for the bit-error probability in lightwave systems with optical amplifiers," *IEEE/OSA Journal of Lightwave Technology*, vol. 8, no. 12, pp. 1816–1823, December 1990.
5. P. C. Becker *et al.*, "Erbium-doped fiber amplifiers fundamentals and technology," in *Optics and Photonics*. NY Academic Press, 1999.
6. "Architecture of Optical Transport Networks," ITU-T Rec. G.872, November 2001.
7. C. Pinart and G. Junyent, "The INIM system: in-service non-intrusive monitoring for QoS enabled transparent WDM," *to appear in the IEEE JSTQE*, 2006.
8. L. B. James, A. W. Moore, and M. Glick, "Structured errors in optical Gigabit Ethernet," in *Proc. Passive and Active Measurement Workshop (PAM)*, April 2004.

Ontology-Based Policy Refinement Using SWRL Rules for Management Information Definitions in OWL

Antonio Guerrero[1], Víctor A. Villagrá[1], Jorge E. López de Vergara[2], Alfonso Sánchez-Macián[1], and Julio Berrocal[1]

[1] Dpto. de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid
[2] Dpto. de Ingeniería Informática, Universidad Autónoma de Madrid
antonio.guerrerocasteleiro@telefonica.es, villagra@dit.upm.es, jorge.lopez_vergara@uam.es, aasmp@dit.upm.es, berrocal@dit.upm.es

Abstract. The goal of ontology-based management is to improve the manageability of network resources through the application of formal ontologies. Prior research work has studied their application to represent the management information definitions, the mapping and merging processes to obtain a semantic integration of those definitions, and the representation of behaviour and policy definitions. Using ontologies allows the additional advantage of integrating, in the same semantic manager, business and service level ontologies with the network management ontology, in a framework for automated management. This integration allows for policy refinement and interoperation between high level policies and low level policies.

1 Introduction

Network administrators need more intelligent management systems that hide the underlying complexity of the network, allowing them to manage the infrastructure at an abstract level, focusing on what the expected behaviour should be, instead of on how to specifically achieve it. In this context, Policy-Based Network Management (PBNM) [1] proposes the use of policies to administer, manage, and control network resources, in such a way that they can be centrally defined and applied to large numbers of devices uniformly. In [2] Strassner depicts the "Policy Continuum", where policies can be defined at several layers with different levels of abstraction. This layering should allow network administrators to manage their systems at a higher level of abstraction than the mere technology configuration, therefore hiding the complexity from the administrator.

This paper presents a generic ontology-based approach to bind the behaviour specified at higher levels of abstraction to the expected behaviour at the network level, in such a way that an ontology reasoner can dynamically perform this High Level (HL) to Low Level (LL) refinement process at run-time. The next sections describe the semantic management framework within which this work is presented, and how the policy refinement process can be accomplished. Then, a simplified policy refinement example will be used to illustrate the mechanisms being presented.

2 Semantic Management

The ontology-based semantic management framework [3] proposes a single manager working with a unique information model, which integrates all the different definitions of the managed resources, taking into account the semantic aspects of those definitions (i.e. their meaning). In [4] it is shown how to merge and map management definitions from different domains into a Common Management Ontology. A semantic manager could then apply generic policies for all the network resources, independently of the management models in which they are originally defined.

OWL [5], the Web Ontology Language, is proposed as the language for policy and management definitions, since it contains all the necessary constructors to formally describe most of the information management definitions [6]. This semantic approach allows the integration, in the same unified management information model, of the behaviour definitions and policies for the managed resources, which can also be expressed in OWL using the SWRL language [7], as shown in [8].

A comparison of other Semantic Web policy languages is presented in [9], from a PBM point of view, stating that the possibility to represent entities and behaviours at multiple levels of abstraction makes ontology frameworks adequate to deal with several kinds of contexts at different level of specifications. The advantages of semantic policy frameworks are analysed in [10], stating that semantic approaches using RDF/OWL as standards for policy representation enable runtime extensibility and adaptability of the system, as well as the ability to work with policies relating to entities described at different levels of abstraction. The use of ontology-based PBM to provide dynamically adaptive network management solutions is also proposed in [11].

3 Ontology-Based Policy Refinement

The proposed semantic manager can therefore work with ontologies and policies defined at different abstraction levels, which allows facing one classical problem in the PBNM area: policy refinement. Policy refinement is concerned with the process of mapping a set of HL policies to a set of LL policies [1]. Most approaches, such as in [12] and [13], attempt this decomposition of HL policies relevant to a composite system into a set of policies that are executed in its constituent parts to implement the behaviour intended by the overall higher level policies. In contrast to refinement, [14] introduces the concept of *Policy Interoperability*. While refinement is concerned with the unidirectional mapping HL →LL, interoperability is the bi-directional mapping HL↔LL. The purpose of this interoperability mapping is to allow LL policies at runtime to dynamically refer to their HL parents as the need arises.

The approach being presented can be summarized in the following three steps:

1) First, we have OWL ontologies both for the upper domain an the lower domain. Definitions of HL and LL policies could be included, as shown in [8].
2) Relating HL ontologies to LL ontologies can be achieved in the OWL ontology language by means of meaningful OWL relationships between HL and LL classes. These will be referred to as *Interoperability Relationships*.
3) Finally, translation SWRL rules can be used to make the semantic manager able to derive the necessary information translations in order to: 1) populate the higher level

with data useful for this layer, hiding the complexity of the data at the lower level, and 2) add data to the lower level based on the information from the upper layer. The following is a generic example of 1) in SWRL logic syntax:

```
LLproperty1(?LLclassYindividual) ^
InteroperablityRelationship1(?LLclassYindividual,
?HLclassAindividual) => HLproperty1(?HLclassAindividual)
```

More complex conditions combining classes from both layers could be expressed.

With the model and the SWRL rules programmed, the manager will be able to perform this bi-directional information mapping at run time, in such a way that changes in the HL data affect the LL data and vice versa. This way, policies defined at the HL layer can govern policies at the LL layer, achieving dynamic policy interoperability.

4 Proof of Concept Use Case: Backup for DSL Premium Lines

The scenario for the use case is an Internet Network Access service offered by a Service Provider for thousands of users. The service is supported by an IP backbone and an ATM access network. Each subscriber's modem-router is connected to its corresponding Broadband Remote Access Server (BRAS), through an ATM circuit, that runs over the telephone line and enters the ATM network through the DSLAM.

Since DSL circuits run over the telephone lines (POTS or ISDN), which can also be used for dial-up Internet access, the Service Provider wants to offer a backup service for some of his DSL subscribers. For this matter he has installed a Remote Access Server (RAS) that will accept incoming telephone connections from the subscribers' modem-routers. However, he only wants subscribers with a PREMIUM contract to make use of this backup network infrastructure, so he has installed a RADIUS server in order to authorize or deny access through the RAS.

The ontology model for this scenario, including the SWRL rules, has been defined in the namespace http://www.dit.upm.es/jlopez/geseman/policy.owl#, and a simulation has been implemented in Bossam [15], a Rule/OWL reasoner.

4.1 HL and LL Ontologies in OWL

At the network level we would have an Integrated Network Management Ontology, such as the Common Management Ontology proposed in [3]. For simplicity purposes, we have restricted the information used in this example to what is strictly needed. The RDF graph [16] in Fig. 1 shows the chosen HL and LL classes and properties.

In the HL specifications, we can also define the HL policies, as shown in [8]: If a subscriber has a PREMIUM service contract, and his DSL service is down, then he is allowed to use the telephone backup access service. For this HL policy, we used the following SWRL rule:

```
ServiceContract(?subscriber, ?contract) ^
swrlb:equal(?contract, "PREMIUM") ^
DSLServiceStatus(?subscriber, ?status)  ^
swrlb:equal(?status, "NOT OK")
=> BackupAllowedSubscriber(?subscriber, "YES")
```

Other two HL rules were used in the simulation to set the value of the *BackupAllowedSubscriber* to “NO” when appropriate.

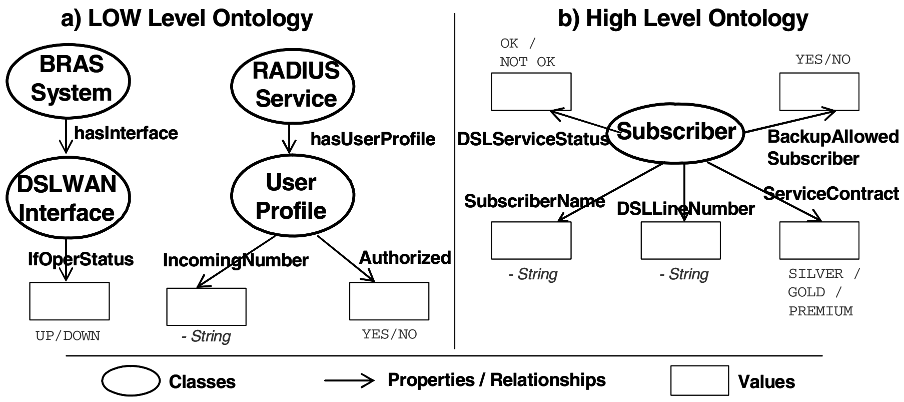


Fig. 1. RDF graph representation of High and Low Level ontologies

4.2 Interoperability Relationships

New relations in order to bind the HL and LL representations are required:

- A *Suscriber ⇔ DSLWANInterface* binding: the *hasWANInterface* relationship. For this binding, all interface data for all users will come from the provisioning system in this example, so it will be available in the semantic manager’s database of facts.
- A RADIUS *UserProfile ⇔ Subscriber* binding: the *relatesToSubscriber* relationship. This binding will take place at run time. It will be inferred by the semantic management system whenever an incoming call enters the RAS.

4.3 Translation SWRL Rules

Rules 1 an 2: *DSLServiceStatus* should be “OK” if the *ifOperStatus* of the subscriber’s WAN interface is “UP”, and “NOT OK” otherwise. This is the representation of the first rule in SWRL logic syntax:

```
hasWANInterface(?subscriber, ?wanif) ^ ifOperStatus(?wanif,
?operstatus) ^ swrlb:equal(?operstatus, "UP")
=> DSLServiceStatus(?subscriber, "OK")
```

This is an example of setting HL information from LL information.
Rule 3: A *UserProfile* relates to a specific *Subscriber* if the *IncomingNumber* of the profile matches the subscriber’s *DSLLineNumber*. This rule is an example of relating HL information to LL information. Expressed in SWRL logic syntax:

```
swrlb:equal(IncomingNumber(?userprofile),
DSLLineNumber(?subscriber))
=> relatesToSubscriber(?userprofile, ?subscriber)
```

Rules 4 and 5 (SWRL not shown): The value for the *Authorized* property of *UserProfile* should be "YES" if subscriber is allowed to use the backup service, and "NO" otherwise. These rules are examples of setting LL information from HL information.

4.4 Overview of the Backup for DSL Lines Use Case

The semantic manager holds the database of facts, with all of the facts and axioms of the ontology model, including HL Policies, HL facts (subscribers' service contracts and DSL line numbers), LL facts (operational status for all DSL WAN Interfaces), HL to LL bindings, and Translation SWRL Rules. With all this information, the ontology manager – having an inference engine – is able to act as a PDP (*Policy Decision Point*), answering the query on whether an incoming call, identified by its telephone number, should be allowed to access the network through the RAS. For a generic implementation, the RDF query in RDQL (RDF Query Language), would be

```
SELECT ?auth   WHERE (dit:<subscriber> dit:Authorized ?auth)
USING dit FOR <http://www.dit.upm.es/jlopez/geseman/policy.owl#>
```

which returns a value of "*YES*" or "*NO*" for the *auth* variable depending on the instance of <*subscriber*> entered and the simulation data. The RADIUS service would finally notify the allowance or denial of the incoming call to the RAS (*Policy Enforcement Point*).

5 Conclusion and Further Work

A general purpose ontology reasoner can work with HL and LL ontologies, being completely independent of their abstraction levels, and allowing for interoperability, whereas some other expert systems and policy languages usually have a more specific orientation. The ontology reasoner, having an inference engine, will be able to understand the model, work with the network data, and enforce the expected behaviour, therefore becoming an implementation of a management system.

The present work presents an approach on how ontology representation could be used for dynamic policy interoperability between HL business rules and LL network policies, while maintaining the separation of concepts of HL and LL information.

In the use case presented as a proof of concept, a change at the network level, such as when a DSL connection goes down, affects the expected behaviour of the Remote Access Server, also at the network level. This LL behaviour is governed by authorization HL business policies, in a dynamic and bi-directional refinement cycle.

Unlike other methods such as those presented in [12] and [13], this approach does not attempt to directly translate policies from the upper level into a set of policies or configuration commands at the lower level. Also, it is not restricted to service oriented architectures as in [11], in which behaviour defined for a certain HL service affects the behaviour of the LL services in which the former is decomposed. It also presents the semantic web advantages for network management about working with data distributed over different systems with heterogeneous RDF-based semantics [11].

The possibility to represent and work with meaningful and reusable interrelations between different abstraction levels could be useful for other specific purposes of ontology-based network management. Areas for further work include its application to event enrichment, service composition, and event correlation.

References

1. A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, S. Waldbusser: Terminology for Policy-Based Management. IETF Request For Comments 3198 (2001)
2. J. Strassner: Policy-Based Network Management – Solutions for the Next Generation. Morgan Kauffman (2003)
3. J. E. López de Vergara, V. A. Villagrá, J. I. Asensio, J. Berrocal: Ontologies: Giving Semantics to Network Management Models. IEEE Network, Vol. 17, Issue 3 (2003) 15-21.
4. J. E. López de Vergara, V. A. Villagrá, J. Berrocal: Benefits of Using Ontologies in the Management of High Speed Networks. Proc. 7th IEEE Intl. Conf. on High Speed Networks and Multimedia Communications (HSNMC'04), LNCS 3079, Toulouse, France (June 2004) 1007-1018.
5. M. K. Smith, C. Welty, D. L. McGuinness: OWL Web Ontology Language Guide. W3C Recommendation, (February 2004)
6. J. E. López de Vergara, V. A. Villagrá, J. Berrocal: Applying the Web Ontology Language to management information definitions. IEEE Communications Magazine, Vol. 42, Issue 7 (2004) 68-74.
7. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission (21 May 2004)
8. A. Guerrero, V. Villagrá, J. E. López de Vergara, J. Berrocal: Ontology-Based Integration of Management Behaviour and Information Definitions Using SWRL and OWL. Proc. 16th IFIP/IEEE Intl. Workshop on Distributed Systems: Operation and Management (DSOM'05) , Barcelona, Spain, LNCS 3775 (October 2005) 12-23.
9. G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri1, A. Uszok: Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder. Proc. 2nd Intl. Semantic Web Conference, Sanibel Island, Florida, USA, LNCS 2870 (October 2003) 419-437.
10. F. J. García, G. Martínez, J. A. Botía, A. F. Gómez Skarmeta: Representing Security Policies in Web Information Systems. Proc. Policy Management for the Web (PM4W), 14th Intl. WWW Conference, Chiba, Japan (May 2005)
11. D. Lewis, K. Feeney, K. Carey, T. Tiropanis, S. Courtenage: Semantic-based Policy Engineering for Autonomic Systems. Proc. 1st IFIP Intl. Workshop on Autonomic Communication (WAC 2004), Berlin (October 2004)
12. R. Darimont, A. van Lamsweerde: Formal Refinement Patterns for Goal-Driven Requirements Elaboration. Proc. 4th ACM Symposium on the Foundations of Software Eng. (FSE4) (1996) 179-190.
13. A. Bandara, E. Lupu, J. Moffet, A. Russo: A Goal-based Approach to Policy Refinement. Proc. 5th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2004)
14. S. Magrath, R. Braun, F. Cuervo: Policy Interoperability and Network Autonomics. Proc. 1st IFIP Int. Workshop on Autonomic Communication (WAC 2004), Berlin (October 2004)
15. Minsu Jang, Joo-chan Sohn: Bossam: an extended rule engine for the web. Proc. 3rd RuleML Intl. Workshop (RuleML 2004), LNCS Vol. 3323, (November 2004) 128-138.
16. F. Manola, E. Miller: RDF Primer, W3C Recommendation (10 February 2004)

Reconfiguring Self-stabilizing Publish/Subscribe Systems

Michael A. Jaeger[⋆], Gero Mühl[⋆⋆], Matthias Werner, and Helge Parzyjegla[⋆⋆⋆]

Communication and Operating Systems Group
Berlin University of Technology
Einsteinufer 17, 10587 Berlin, Germany
{michael.jaeger, g_muehl, m_werner, parzyjegla}@acm.org

Abstract. Recent work on self-stabilizing routing in publish/subscribe systems showed that it is feasible to automate reconfigurations in case of faults by enabling the system to recover from arbitrary transient faults. In this paper, we discuss how to incorporate planned reconfigurations of the broker topology into self-stabilizing publish/subscribe systems without service interruption. We present an algorithm that uses a coloring mechanism to enable the system to be automatically switched from one system configuration to another. The colors thereby synchronize the broker overlay and the publish/subscribe routing layer.

1 Introduction

A publish/subscribe (pub/sub) system consists of brokers and clients. *Brokers* connect to other brokers to form an overlay network and to provide the *event notification service*. *Clients* connect to one broker and *publish notifications* or *subscribe to filters*. The broker overlay network routes published notifications to all brokers with clients that are currently subscribed to a matching filter.

Recent work on fault tolerance in the field of pub/sub middleware has shown that self-stabilization is feasible on the pub/sub routing layer [6] (for the sake of readability, we will use "routing" in the following when we actually mean "pub/sub routing"). Self-stabilization is an elegant mechanism for gaining fault tolerance. However, the solutions presented for pub/sub routing do not yet explicitly deal with reconfiguration [8,10], although managing these systems has to include reconfiguration of the broker overlay topology. In many self-stabilizing systems, reconfigurations are treated as faults and the system tries to "recover" from them. In contrast to this, our approach is to hold a "shadow" broker overlay topology that has already implemented the reconfiguration, to subsequently build up "shadow" routing tables on the pub/sub routing layer, and to finally switch the system atomically from one correct configuration to the next one using a coloring scheme. Thereby, we avoid notification and (un)subscription loss as well as duplication during reconfiguration.

⋆ Funded by Deutsche Telekom Stiftung.
⋆⋆ Funded by Deutsche Telekom.
⋆⋆⋆ Funded by Deutsche Forschungsgemeinschaft (DFG SPP 1183 Organic Computing).

2 Related Work

There are only a few publications in the area of self-stabilization that deal with reconfiguration issues explicitly. Most authors treat them as faults that will eventually stabilize like Dijkstra did in the initial paper on self-stabilization [1]. The concept of *superstabilization*, introduced by Dolev and Herman, has a more differentiated view by explicitly considering a certain class of topological changes [3]. Superstabilizing protocols are self-stabilizing and additionally require that if one change of this class occurs a *passage predicate* holds until the system is stable again. The passage predicate is usually weaker than the correctness predicate but is supposed to be strong enough to be still useful. In contrast to our approach, topological reconfigurations are supposed to happen immediately without any announcement and can thus not be delayed as we assume here. The concept of *fault containment* as described by Nelson [9] and applied to self-stabilizing protocols by Ghosh et al. [5] also tries to maintain service availability by keeping the effects of faults (or reconfigurations) locally bounded. Although fault-containment can dramatically reduce the effects of reconfiguration on the system in whole there is still an interruption of the service—at least in those parts of the system that are directly affected by the reconfiguration.

3 Assumptions and Model

As starting basis, we build on a model for self-stabilizing pub/sub systems developed in previous work [8]. Basically, we assume a hierarchical routing algorithm based on an acyclic broker topology with bidirectional FIFO links connecting individual nodes. Furthermore, there is an upper bound η on the number of brokers as well as a dedicated *root broker* R, which is globally known within the system. Considering self-stabilization, we require that the brokers' routing tables can be rebuilt from an *initial routing configuration*, which is empty for many routing algorithms [7], and that the routing algorithm bases its routing decisions solely on the contents of the routing table and the notification to forward.

Such a pub/sub system works *correctly*, if it meets the following two requirements [8]: (i) every client receives only the published notifications it has subscribed for (without duplicates) and (ii) every subscription becomes active after finite time, from which on the client receives every published notification matching its subscription until it unsubscribes.

A pub/sub system is called *self-stabilizing*, if started in an arbitrary state, it eventually begins to satisfy its specified behavior provided that no faults occur for a sufficient long time. A *fault* may lead to arbitrary perturbations of any variable stored in RAM as well as removed, manipulated, and inserted messages. Links may go down and come up, processes may crash and restart due to faults.

To guarantee persistence in spite of arbitrary memory faults, we assume that all algorithms used are stored in non-perturbable ROM. Additionally, we treat the reference to the root broker as well as the initial routing configuration as an intrinsic part of the respective algorithm itself and include them in ROM, too.

To maintain self-stabilization in case of crashed processes, the root broker R can be implemented in a self-stabilizing fashion using a root group [4].

While faults happen suddenly and may lead to abrupt changes in the broker topology, a reconfiguration is a *cooperative* process that is usually planned in advance and needs some time to take effect. More precisely, a *reconfiguration* is a change of the broker overlay topology, including leaf broker removals, additions, and link replacements, that can be delayed for a finite time.

Since reconfigurations affect several algorithm layers simultaneously, all actions carried out must be synchronized to meet the system's correctness requirements and to reach atomicity. A major challenge is to integrate the individual self-stabilizing algorithms of each layer into the whole reconfiguration process.

4 Layered Self-stabilization

Systems that are layered can be made self-stabilizing by making all layers individually self-stabilizing. This transparent stacking of self-stabilizing layers is a standard technique which is referred to as *fair composition* [2]. It is easy to combine self-stabilizing algorithms this way to create a new and more powerful self-stabilizing mechanism as long as no cyclic dependencies exist among the layers. Taking this approach, it is sensible to layer self-stabilizing routing in pub/sub systems on top of a broker topology that employs a self-stabilizing tree algorithm like the ones given in literature. However, this approach has its drawbacks because a reconfiguration on the broker overlay layer may be handled as a fault on the pub/sub layer when the routing table entries are not consistent with the new topology anymore. Additionally, most self-stabilizing tree algorithms impose a specific structure on the topology that is, for example, dependent on the IDs of the nodes. As a consequence, a topological reconfiguration of the self-stabilizing pub/sub system might result in a service interruption like missed notifications or control messages (subscriptions and unsubscriptions).

Our approach in the following is to realize a self-stabilizing overlay topology that maintains an arbitrary tree structure and to layer self-stabilizing routing on top of it in a way, such that reconfigurations of the overlay topology can be processed without service interruption. Two problems have to be tackled to solve this problem: (i) designing a self-stabilizing broker overlay topology that does not necessarily impose a certain structure on the resulting tree and (ii) coupling the self-stabilizing mechanisms on the overlay and the routing layer to allow for atomic topology switches without loss of messages.

Coloring Scheme. The coloring scheme synchronizes reconfigurations on the overlay layer with the routing layer. Therefore, selected data structures are marked with a *color* attribute. On the overlay topology layer this concerns the child and parent broker pointers ($\mathcal{C}$ and $\mathcal{P}$, respectively), while on the routing layer the routing entries are affected. To allow atomic switches between different colors, every broker maintains data structures for three different colors that can be accessed on both layers: the color c^{cur} that is currently used, the color c^{old}

that has been used last, and the color c^{new} that will be used when the color changes for the next time. These colors are rotated regularly. The reason why we need three different colors is due to the communication and processing delay in the network. If the value of c^{cur} becomes the value of c^{old}, for example, there may still be messages on the network that are colored with c^{old}. To be able to deliver these messages, the topology for c^{old} has to be kept long enough. For a better understanding, we assume in the following that the routing entries are stored in separate routing tables T for each color although a tag on each entry suffices in the implementation.

It is the task of the root broker R to regularly recolor all brokers in the tree. To accomplish this, a timeout runs on every broker that triggers different actions on R and on each broker $B \neq R$. On a timeout, R resets its timer, rotates its colors and subsequently initializes the child broker pointers $\mathcal{C}^{c^{\text{new}}}$ and the parent broker pointer $\mathcal{P}^{c^{\text{new}}}$ with the respective values colored with c^{cur} (which has been c^{new} before the timeout). The routing table $T^{c^{\text{new}}}$ is initialized with the initial routing configuration. Then, it disseminates the new color in a *recolor message* REC_{msg} to all child brokers stored in $\mathcal{C}^{c^{\text{cur}}}$, if they are still alive as indicated by a flag that was set when the child broker acknowledged the previous REC_{msg}. For every other broker $B \neq R$ a timeout is viewed as a fault and hence B tries to reconnect to the tree (as part of the self-stabilizing overlay topology). Reconfigurations are stored in REC_{msg} and handled as described later in a separate section. When B receives a recolor message, it resets its timer, replies with an *acknowledge message*, rotates its colors, initializes its pointers like R, and forwards the message to its child brokers. The broker accepts the recolor message only if it has been sent by the broker $\mathcal{P}^{c^{\text{new}}}$ points to and if the new color $m.c$ stored in the message is different from the color stored in c^{new}. This test is needed to detect cycles that may result from faults.

Self-Stabilizing Broker Overlay Topology. The self-stabilizing mechanism on the broker overlay network is based on timeouts regarding recolor messages as described above. Recolor messages are forwarded recursively down the tree, the last leaf broker receives the message at the latest after time $h \cdot \delta_{\max}$, where h is the height of the tree and $\delta_{\max}$ is the maximum delay for processing and sending a message to a child broker. As the tree may degenerate arbitrarily h can be at most equal to the maximum number of brokers η in the system (which we assume is known and stored in ROM). Given that the timeout on R occurs every time ξ, a timeout $\xi' = \xi + h \cdot \delta_{\max}$ is necessary on every broker B distinct from R, which is resetted everytime a new recolor message is received from its parent broker. When $B \neq R$ runs into a timeout, it took more than ξ' to receive the next recolor message after the last one. This can only be due to a fault, since forwarding a message from R to B cannot take more than $h \cdot \delta_{\max}$. In this case, B contacts R to rejoin the tree. There are many ways to find a new parent broker for B depending on the topology requirements. One is to look for an arbitrary broker that has less than b child brokers down the tree and use it as a new parent for a requesting broker. This way, the broker is integrated as a leaf into the tree and the degree of a the broker topology can be maintained. The broker overlay is in a *correct state* if the parent and child broker relation between every broker in the system is consistent for the data structures colored with the values of c^{old} and c^{cur} at R and the tree that is defined by $\mathcal{P}^{c^{\text{old}}}$ and $\mathcal{C}^{c^{\text{old}}}$, and $\mathcal{P}^{c^{\text{cur}}}$ and $\mathcal{C}^{c^{\text{cur}}}$ respectively, is not partitioned. The value of $\mathcal{C}^{c^{\text{new}}}$ and $\mathcal{P}^{c^{\text{new}}}$ is treated differently as explained in the next subsection about reconfiguration.

Reconfiguration. Whenever a leaf broker wants to join or leave the overlay network or a link has to be replaced by another one, the topology of the broker network changes. When a reconfiguration should be implemented, the intended changes are sent to R, which collects them in the set $\mathfrak{R}$ and disseminates them in the next recolor message. Every broker that receives a recolor message carrying reconfiguration data that affects it, implements the change into its $\mathcal{P}^{c^{\text{new}}}$ and $\mathcal{C}^{c^{\text{new}}}$ pointers. The recolor message serves as a synchronizer to prevent race conditions when switching from one topology to another. Recolor messages are routed using $\mathcal{C}^{c^{\text{cur}}}$ of every broker B that receives a recolor message (where c^{cur} equals c^{new} before recoloring). Thus, reconfigurations take two recolor messages to become active: one to disseminate the reconfiguration and one to activate it.

As mentioned earlier, a change in the topology may imply a change in the routing tables on the pub/sub routing layer. As the routing tables are regularly rebuilt from an initial routing configuration the reconfiguration of the overlay topology can be incorporated by delaying the switch to the new topology in $\mathcal{P}^{c^{\text{new}}}$ and $\mathcal{C}^{c^{\text{new}}}$ long enough, such that they have been rebuilt completely.

Self-Stabilizing Routing. Recolor messages are used on the topology layer to trigger timeouts and coordinate reconfigurations. Therefore, three different topologies are held in form of colored parent/child pointers. On the routing layer, the color is used for two different purposes: (i) to rebuild the routing tables periodically and (ii) to avoid notification loss and duplicates.

It is necessary to periodically rebuild the routing tables as we assume that they can be perturbed arbitrarily. Therefore, we rely on the leasing mechanism described earlier [8]: clients regularly refresh their subscriptions and brokers use a second chance algorithm to remove stale entries from their routing tables. To incorporate reconfigurations into this mechanism, we demand that control messages are colored with c^{new}, while notifications are colored with c^{cur}. Notifications and control messages are then forwarded and applied to the routing tables $T^{c^{\text{cur}}}$ and $T^{c^{\text{new}}}$, respectively. Thereby, we ensure that notifications will be routed over the topology, the publishing broker belonged to at publishing time. This way, we prevent duplicates, i.e., notifications sent multiple times to the same broker. The second chance algorithm is implemented through rotating the colors and initializing $T^{c^{\text{new}}}$ with a legal initial routing configuration.

5 Summary

We presented an algorithm that allows self-stabilizing pub/sub systems to be reconfigured while maintaining service availability. To achieve this, we use the color attribute to synchronize the self-stabilizing broker overlay and the self-stabilizing pub/sub routing layer. We connect the different layers such that it is possible to switch the topology atomically without losing or duplicating messages. The presumption is that reconfigurations can be delayed a bounded time before becoming active (e.g., before a broker is removed). We consider this "co-operative" behavior as the main difference between a fault and a reconfiguration. Our work is a necessary prerequisite to combine the self-stabilizing routing layer with an adaptive reconfiguration mechanism that runs on top of the pub/sub layer and issues reconfiguration stimuli that are implemented by the lower layers as described in this paper. Hence we come one step closer to fault-tolerant and adaptive publish/subscribe systems. However, the mechanism we described is not limited to self-stabilizing pub/sub. It is a general principle that can be used to realize reconfigurations in arbitrary layered self-stabilizing systems.

References

1. E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974.
2. S. Dolev. *Self-Stabilization.* MIT Press, 2000.
3. S. Dolev and T. Herman. Superstabilizing protocols for dynamic distributed systems. *Chicago Journal of Theoretical Computer Science*, 4, Dec. 1997. Special Issue on Self-Stabilization.
4. S. Dolev and R. I. Kat. Hypertree for self-stabilizing peer-to-peer systems. In *Network Computing and Applications (NCA 2004). Proceedings. Third IEEE International Symposium on*, pages 25–32, Washington, DC, USA, 2004. IEEE.
5. S. Ghosh, A. Gupta, T. Herman, and S. Pemmaraju. Fault-containing self-stabilizing algorithms. In *Proceedings of the Fifteenth Annual ACM Symposium of Distributed Computing (PODC96)*, pages 45–54. ACM, ACM, 1996.
6. M. A. Jaeger and G. Mühl. Stochastic analysis and comparison of self-stabilizing routing algorithms for publish/subscribe systems. In G. F. Riley, R. Fujimoto, and H. Karatza, editors, *The 13th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2005)*, pages 471–479, Atlanta, Georgia, USA, Sept. 2005. IEEE Press.
7. G. Mühl. *Large-Scale Content-Based Publish/Subscribe Systems.* PhD thesis, Darmstadt University of Technology, Sept. 2002.
8. G. Mühl, M. A. Jaeger, K. Herrmann, T. Weis, L. Fiege, and A. Ulbrich. Self-stabilizing publish/subscribe systems: Algorithms and evaluation. In J. C. Cunha and P. D. Medeiros, editors, *Proceedings of the 11th European Conference on Parallel Processing (Euro-Par 2005)*, volume 3648 of *Lecture Notes in Computer Science (LNCS)*, pages 664–674, Lisboa, Portugal, Aug. 2005. Springer.
9. V. P. Nelson. Fault-tolerant computing: Fundamental concepts. *Computer*, 23(7):19–25, 1990.
10. Z. Shen and S. Tirthapura. Self-stabilizing routing in publish-subscribe systems. In *3rd International Workshop on Distributed Event-Based Systems (DEBS'04)*, pages 92–97, Edinburgh, Scotland, UK, May 2004. IEE.

Policy and Profile: Enabling Self-knowledge for Autonomic Systems

Ray Carroll[1], John Strassner[2], Greg Cox[2], and Sven van der Meer[1]

[1] TSSG, Waterford Institute of Technology, Waterford, Ireland
{rcarroll, vdmeer}@tssg.org
[2] Motorola Labs, Schaumburg, IL, USA
{john.strassner, greg.cox}@motorola.com

Abstract. The standard definition of autonomics is that of self-governance, including such properties as self-configuring, self-healing and self-optimizing. To really enable self-anything, however, we must first deliver another 'self-' property - self-knowledge. We define self-knowledge as information about a system enabling it to reason on its own capabilities and actions. This knowledge can come in many forms but we propose that there are essentially two key elements: knowledge of the individual parts of the system, and knowledge about the rules that determine the interaction of these system components. This paper presents a model describing self-knowledge, with policy for defining rules and profiles to express the individual entities knowledge.

1 Introduction

Autonomic management [1] aims to ease system management by abstracting complexity and making common management tasks the responsibility of the system, rather than the administrator. A critical objective of autonomic systems is the need to be able to adapt to changes in the managed environment. For example, an autonomic *network* [2] would adapt it's services and resources in accordance with changing environmental conditions and user needs. In order for any system to be capable of this sort of autonomic management, it must be able to understand its own component parts and the combined effect of these parts. In essence, autonomics is about self-knowledge [3] and the usage of this knowledge to determine an appropriate action. It is our supposition that self-knowledge is essentially a function of two factors.

1. Knowledge of the rules that govern the system: high-level business rules which determine the overall goal of the system and specific rules (scoped by higher-level rules) that govern the interaction of system parts for specific situations.
2. Knowledge of the system itself, i.e. all relevant managed elements within the system and all their relevant possible roles/functions and data.

In this paper we present a model describing the factors listed above in the form of policy (1), profile and roles (2) and the relationships between these. Policies are rules that govern a system and its components, and hence represent information about how the system acts. The combination of profile and roles provide individual entity data and behavioural information. Section 2 describes our policy framework and section 3 explains our profile framework. Section 4 then describes our model of how policy, profile and roles interact to facilitate self-knowledge for autonomic network management systems. Section 5 then presents our conclusions and future work.

2 Policy

As mentioned in the introduction section, policy rules are used to govern an autonomic system. In the DEN-ng [4] model policy is realized as an Event, Condition, Action (ECA) triplet, having the semantics: "ON event, evaluate condition clause, THEN execute appropriate actions in the action clause". One or more of a PolicyRule's PolicyEvents trigger the evaluation of a PolicyRule. A PolicyEvent may contain flexibly defined combinations of events (PolicyEventComposite) or individual events (PolicyEventAtomic). As such, a PolicyRule may be triggered on a combination or sequence of events.

When a PolicyRule is triggered by PolicyEvents, evaluation of a PolicyRule's PolicyConditions occurs. Similar to PolicyEvents, PolicyConditions also can include combinations of conditions as PolicyConditionComposite or individual conditions as PolicyConditionAtomic. Further, an attribute of PolicyConditionComposite allows for specification of whether the composite condition is expressed in Conjunctive Normal Form or Disjunctive Normal Form. These features and others in the DEN-ng policy model allow for the expression of complex PolicyConditions in a PolicyRule.

When a PolicyEvent triggers evaluation of the PolicyConditions, then one or more of a PolicyRule's PolicyActions can occur. There are two types of actions: pass actions are invoked if the condition is TRUE, and fail actions are invoked if the condition is FALSE. Like before PolicyActions can include combinations of PolicyActionComposite objects and/or PolicyActionAtomic objects. As such, complex actions and sequences of actions can be expressed using the DEN-ng policy model. Taken together, all these features provide a rich expression of policy needed to enable knowledge of the rules that govern the interaction of system parts in today's complex systems. The need to accommodate multiple views as described in the DEN-ng Policy Continuum [4] further motivates this rich means of modelling policy, acknowledging that the various stakeholders in a complex system have different views of policy and degrees of abstraction in policy expression. DEN-ng defines 5 views: Business, System, Network, Device and Instance. Policies become more specific and increase in technical detail as one moves from the Business to the Instance View. Other policy models exist in the art (e.g., Ponder policy specification language) and are discussed in [4]. However, these alternative policy models tend to be less general than the DEN-ng ECA approach, often tailored to a specific use. As such, this work focuses on the DEN-ng approach employing ECA rules.

3 Roles and Profiles

We initially listed two items that are important for self-knowledge in any system, where Policy provides the first of these. The second was knowledge about the individual components of the system. Our approach to modelling this information is to develop a generic framework that allows us to define entity information in a standard yet flexible way. The principle aim is to allow entities have different sets of information as per their functionality and also to allow information from various sources be associated to an entity. As such we propose the concepts of roles and profiles.

3.1 Roles

Typically, the behaviour of a managed entity is represented by the actual methods and properties of that entity. However, methods and properties alone are not enough to enable self-management as they do not allow us to fully understand an entities function or how it interacts with other entities. In many cases, an entity may have different attributes, tasks or functions that depend on the current situation, and so will use different sets of attributes and methods to execute a task. This enables the entity to adapt to a particular context. The DEN-ng model uses the role-object pattern [5] and we introduce the idea of **Role** to abstract this, so that the different situational requirements are not dependent on individual entities. This enables the model of the entity to be separated from the model of the functions that the entity takes on, and is seen in Figures 1 & 2. Here we see that an Entity (e.g. Organisation or Individual) can aggregate zero or more EntityRoles. This enables the Entity to take on the characteristics of two different entities (e.g. Vendor and Service Provider) without changing the attributes of the Entity directly. This reflects the real world as the Entity itself did not change, only the role that it was playing at a given time.

Role is a well established concept in terms of access control [6] and has also received attention in context–aware and ubiquitous systems [7, 8]. For us, role presents a means to specify what information is relevant based on the current function it is trying to fulfil. Thus, roles present an extensible means for us to introduce contextual characteristics and behaviours, modelled as classes, into the model. Roles also allow separate sources of data that together prove more useful to be grouped together. Fig. 1 shows that EntityRole has an aggregation relationship to Data. This enables data to be associated to an entity via its roles, making pertinent information available based on the set of active roles of an entity.

3.2 Profiles

There are many initiatives that propose the use of profiles at some level. However, most of these are focused on some very specific purpose. Our aim is not to specify particular profile data models, but rather to develop a profiling framework for integrating these disparate sources of information. We believe this will not only increase system knowledge, but also improve accessibility, reusability and overall usefulness of an entity's information. We define a new variant of the role object pattern [5], where the Role object itself aggregates new information. We call this new information Data and use the composite pattern to define appropriate subclasses of data. This enables additional data to be associated to a particular role, instead of embedding the data in a role. We then specialise data to suit the needs of a Profile (Figure 1). This enables each set of profile information to be defined independently and attached to a role at runtime. A Profile is a container for data and we specialise this to EntityProfile, for data about a specific entity. We then use a Composite Pattern [9] to separate EntityProfile into a container (EntityProfileComposite) and component

(EntityProfileComponent) enable hierarchies of Profiles to be built. This allows us to provide a single profile that can contain many profile components and enhances the approach specified in initiatives like 3GPP Generic User Profile [10].

An EntityProfileComponent is also a container for ProfileData. ProfileData is subclassed from EntityData, which is a generic class for any data related to some entity. This structure provides future flexibility and extensibility for our design. ProfileData is the actual data about an entity that composes a Profile. Again, the composite pattern is used so that ProfileData can be either atomic or composed of other data (i.e. nested data values). Figure 2 gives a simple example of a user entity and their associated roles and profiles.

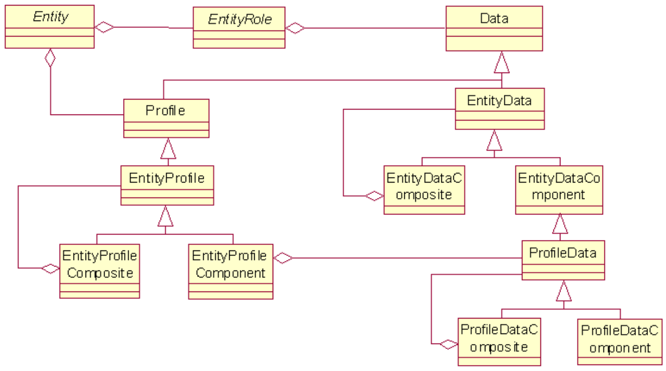


Fig. 1. The Data model

The entity JoeBloggs has a role of Employee and a profile jbloggsProfile which aggregates EntityProfileComponents (LIP[11] and CC/PP[12]) and provides management functionality such as adding, removing, updating, querying etc. It is also important that an entity (e.g. JoeBloggs) is associated to its data (e.g. jbloggsProfile) directly (not just via Roles) for more integrated management of profiles. For Joe Bloggs' role of Employee two profile components are relevant (i.e. LIP and CC/PP). This example is based on a user scenario, but the same mechanism may apply to any entity, (e.g. services and resources). This reflects the needs of autonomic systems, which require functionality to change with changing user needs and/or environmental conditions. In our system, we meet this challenge by dynamically instantiating new roles using *policy*. The combination of roles and policy then enable which *profiles* are allowed to be used.

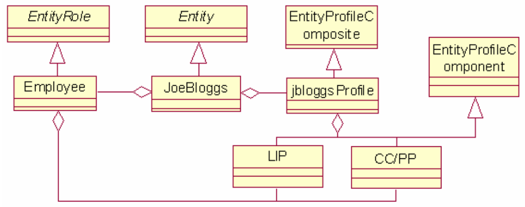


Fig. 2. Example of Entity JoeBloggs with role Employee, profile and profile components

4 Model for Enabling Self-knowledge

In the previous sections we described the individual parts of our proposed system that are core to enabling self-knowledge. However, these parts are not sufficient on their own; rather it is the overall view of how they interact (Figure 3) that makes self-knowledge possible. As the approach to autonomic management that we have adopted is policy-based, policy is a core part of the system and, in effect, governs our system at every level. While each entity provides its own behaviour, policy determines what parts, under what circumstances and to what effect that behaviour is executed. As we can see from Figure 3, Policy **governs** Entity, EntityRole and Data. As regards Entity, Policy will govern specifically their creation, management, and deletion, as well as the addition and removal of roles (and hence profiles) associated with an Entity. Specific semantics of different management functions controlled by Policy for an Entity are defined by the PolicyEntityDetail association class.

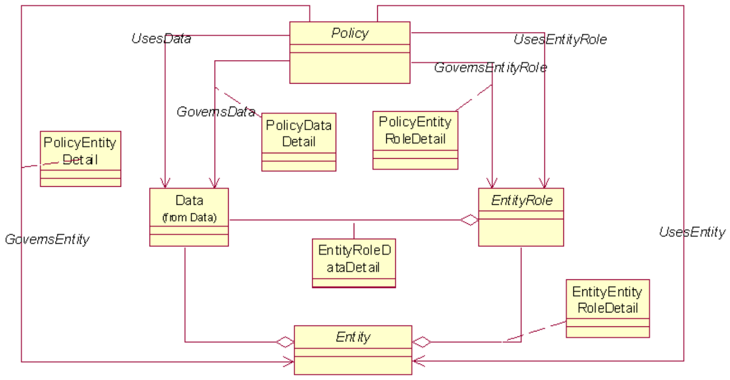


Fig. 3. Overall model of policy, data, entity and role (subclasses omitted for readability)

Since an Entity is largely characterised by its roles, the use of Policy to govern which EntityRoles an Entity can have provides both an abstract view of the Entity's functionality as well as detailed control over the Entitys characteristics and behaviour. Based on the Policies governing the system and knowledge of the environment and users, the EntityRoles relevant to the Entity in question must be determined and enabled or disabled accordingly. This in turn controls the functionality that an Entity has at any given time. The particular semantics of which EntityRole a particular Policy can select is given in the PolicyEntityRoleDetail association class. Multiple EntityRoles can be selected by the same Policy for different reasons and enabling policies to be reused but tailored to the specific needs of a given EntityRole. For example, a device may have many subclasses of the same role (e.g. EdgeDevice, DSLEdgeDevice) but implement that role using very different functionality (e.g. PC v Switch v Router). The Policy should stay the same ("give access") but the particular mechanisms used will vary. This association class enables these details to be captured while keeping the same abstracted pattern. The relationship of EntityRole to Data is characterized by the EntityRoleDataDetails class. This enables specific semantics to be attached to how a given EntityRole uses particular Data.

The third and final governance relationship Policy has is with the Data class. This relationship signifies that Policies will also govern Data; by this we specifically mean that Policy will govern the set of Profiles that an Entity can have. Note, however, that Data is aggregated by EntityRole. In effect, Policy will determine the set of EntityRoles that an Entity can have; based on this, the set of Profiles that are allowed to be used is subsequently determined. As before, the PolicyDataDetail association class enables specific semantics for a given {Data, Policy} combination to be realized. *Policy* also has a number of *Uses* relationships with EntityRole, Data and Entity. These signify that policy will also use these classes in its inherent decision making process.

5 Conclusions and Future Work

This paper has presented a novel design for capturing self-knowledge in autonomic systems using the DEN-ng information model and policy design. In addition, a novel combination of policy, profile and role interaction has been presented: in our system, *Policy* is used to enable or disable the set of *EntityRoles* that a given Entity has; the combination of EntityRole and Policy is in turn used to enable or disable the set of *Profiles* that a given Entity can use. This enables system changes to trigger PolicyEvents, and those PolicyEvents to control the functionality of an Entity (via its roles and Profiles). This enables the autonomic system to (indirectly) use Profile information to control the resources and services that a network provides, as well as those that a user can utilise, as a function of context. Furthermore, it is an extension of the Shared Information/Data model, which is standardised in the TeleManagement Forum (also being considered for standardisation in ETSI TISPAN and ITU-T).

While we have described a model that we feel can enable self-knowledge, we have not yet addressed the issue of how this model can be utilised in a real system to provide autonomic management. A model is a view of the systems underlying data structure and does not provide system behaviour. As such future work will investigate algorithms for processing the components of self-knowledge to truly produce autonomic network behaviour and in line with this also investigate the relationship of this work to context.

References

[1] Kephart, J.O. and Chess, D.M., “The Vision of Autonomic Computing”, IEEE Computer, January 2003, www.research.ibm.com/autonomic/research/papers/

[2] Strassner, J. and Kephart, J., “Autonomic Networks and Systems: Theory and Practice”, NOMS 2006 Tutorial, April 2006

[3] Thomas Cofino et al "Towards Knowledge Management In Autonomic Systems," *iscc*, p. 789, Eighth IEEE Symposium on Computers and Communications, 2003.

[4] Strassner, J., “Policy-based Network Management: Solutions for the Next Generation”, Morgan-Kaufman Publishers. ISBN 1-55860-859-1, (2004)

[5] The Role Object (Design) Pattern. Download PDF from http://www.riehle.org/computer-science-research/1997/plop-1997-role-object.pdf
[6] National Institute of Standards and Technology (NIST), Role Based Access Control. http://csrc.nist.gov/rbac/
[7] Beresnevichiene, Y. "A Role and Context Based Security Model", Technical Report, University of Cambridge, Computer Laboratory January 2003.
[8] Sashima, A. et al. 2006. Toward Role-based Agent Coordination for Mobile and Ubiquitous Services. In *Proceedings of the 20th international Conference on Advanced information Networking and Applications, Volume 02* (April, 2006). AINA. IEEE Computer Society, Washington DC, 262-266.
[9] E. Gamma, et. al., "Design Patterns", pp 163-173
[10] 3GPP, Generic User Profile, http://www.3gpp.org.
[11] IMS Learner Information Package, http://www.imsglobal.org
[12] W3C "CC/PP: Structure and Vocabularies", 2004. http://www.w3c.org

DECA: A Hierarchical Framework for DECentralized Aggregation in DHTs

Marc S. Artigas[1], Pedro García[1], and Antonio F. Skarmeta[2]

[1] Universitat Rovira i Virgili,
Tarragona, Spain
{marc.sanchez, pedro.garcia}@urv.cat
[2] Universidad de Murcia,
Murcia, Spain
skarmeta@dif.um.es

Abstract. As Structured Peer-to-Peer (P2P) Networks become popular, there is an emerging need to monitor continuously the huge number of participants in a robust and scalable manner. To this end, aggregation has emerged as a basis for the self-management of these networks. However, the structured P2P networks lack today of efficient mechanisms for the decentralized computation of these aggregates. In this paper, we propose a hierarchical theoretic model based on Cayley Graphs, which overcomes the requisite to accommodate growth without impacting the efficiency of distributed applications. Also, the paper presents an aggregation protocol that fuses the fault-resilience of gossip algorithms with the scalability of trees. In particular, simulation results show that this algorithm is capable to cope with the distributed and unreliable nature of P2P networks.

Keywords: Structured P2P Networks, DHT, Aggregation, Hierarchical model, Gossip, Cayley Graphs.

1 Introduction

In recent years, Structured Peer-to-Peer (P2P) Networks, which offer an efficient, scalable, resilient and self-organizing substrate for building distributed applications have become widely popular, including Chord[1], CAN[2], Pastry[3] and Tapestry[4]. As these networks grow in popularity, there is an emerging need to collect a variety of statistical information about resources and/or peers to allow primarily individuals and then administrators to perform global control actions without explicit coordination. However, the P2P principles themselves pose a challenge for developing large scale management applications. Particularly, it is apparent that their decentralized nature should not be violated for any reason. Precisely, it is this need of decentralization what do P2P management systems be markedly different from traditional ones. In this line, we believe that the main challenge of P2P paradigm concerning management lies in *developing decentralized architectures capable to support up-to-date techniques without disregarding the symmetric functionality of peers*. To this end, a first natural step consists of enabling AGGREGATION, i.e. the computation of global network parameters through the use of functions MIN, MAX, SUM, COUNT or AVG, which is an efficient technique for provisioning nodes with valuable information.

A research area that can benefit substantially from AGGREGATION is autonomic computing: monitoring is an essential feature of autonomic systems and entails to log statistics about autonomic elements which serve as the basis for self-adaptation; self-healing, self-protection etc. Note that the essence of autonomic computing systems is self-management, the intent of which is to free system administrators from the details of system operation and maintenance.

The focus of this paper is on a decentralized model for computing AGGREGATION functions in Distributed Hash Tables, DHTs. Briefly, a DHT is a decentralized object-location mechanism for P2P systems that manages the distribution of content among a dynamic set of nodes by using a consistent mapping of keys to nodes. The DHT abstraction provides the same functionality as a hash table — associating key-value pairs with physical network nodes rather than hash buckets. They provide the *put*(key, value) and *get*(key) functions to allow DHT members to efficiently store and retrieve stored resources by *name* without using centralized servers. In practice, we consider that a nice AGGREGATION scheme for DHTs must fulfill the following:

i. *Accuracy*: let us consider a request to compute an aggregate function F. Then, "*accuracy*" refers to the maximum allowed error (ζ) for the returned estimation $\hat{F}$. We define accuracy as $(1-\hat{F}/F_{true})$, where F_{true} denotes the true value for function F. Then, if the answer lies in the range $[(1-\zeta)F_{true}, (1+\zeta)F_{true}]$, we say that the aggregation scheme ensures *practical validity*.
ii. *Scalability*: popular P2P networks maintain a large number of participants throughout the time. Consequently, a management protocol must scale to networks of very large size, that is, the load and the traffic generated by message exchanges must be small and evenly distributed.
iii. *Robustness*: the participants of a DHT are expected to be very dynamic. This means that our protocol must adapt gracefully to changes in the overlay, including node and link failures.

In this paper, we propose DECA: a hierarchical management framework capable to augment ordinary DHTs with robust, accurate and scalable AGGREGATION facilities.

Significant contributions distinguish DECA from previous work. These are:

– *Hierarchical P2P-theoretic model* that effortlessly exploits the in-built recursive decomposition of ordinary DHTs. The main reason is that hierarchies are ideal for accommodating growth and isolating faults. To this end, we introduce a powerful tool based on Cayley graphs which converts an overlay — like Chord, Pastry, CAN ... — in a collection of clusters mimicking a hierarchical organization. Hereinafter, we refer to this as function $\mathcal{H}$. Although in [5] we devise a pragmatic technique for constructing hierarchical DHTs, however, $\mathcal{H}$ extents are broader. It endeavors to uncover the hidden hierarchical structure of typical DHTs. Also, we assume that there exists an effective mapping tool that groups topologically close nodes into the same cluster. It is our position to consider that topology concerns are of paramount importance for a practical P2P AGGREGATION service. To the date, we believe that DECA is the first attempt to build an effective management infrastructure built on top of a hierarchical DHT.

– *Efficient* AGGREGATION *protocol* that blends together the *scalability* of *trees* with the *resilience* of *epidemic algorithms*. To compute a global parameter, a system must provide redundancy to ensure practical validity. Node failures must not lead to severely hampering management operation. Hence, a gossip protocol is a better alternative than a single tree to compute function F over the weights of all nodes in the system.

The paper is organized as follows. Section 2 reviews related work. Section 3 describes formally our hierarchical model and AGGREGATION algorithm. Section 4 provides a discussion about function ´H. Section 5 provides Whirl, the Chord instance of our framework. Section 6 describes our simulation results. Finally, Section 7 draws some conclusions.

2 Related Work

Although there exists a large body of literature in the network management area, we believe that DECA is the first attempt to build a P2P distributed AGGREGATION [6, 7] mechanism based on hierarchical distributed hash tables, DHTs.

To the best of our knowledge, there are only a few proposals that try to solve the node aggregation problem in DHTs. In [8, 9, 10], a tree structure is constructed and maintained to propagate aggregates to the root. Except SOMO[10], fault-tolerance is achieved in these approaches by a mechanism that reconstructs the tree after node addition, removal or failure. Ji Li et al. in [8] demonstrated analytically that even with the presence of a refreshing algorithm that corrects failures; tree infrastructures cannot ensure practical validity. Besides, tree-based aggregation schemes share another important difficulty. Since link and node congestion increase as the distance to the root decreases, the scalability of the whole system becomes tightly coupled not only to the nodes capacities, but also to the actual tree organization of participants.

Willow[11] and DASIS[12] built a logical binary tree on top of a P2P system to provide participants with aggregation facilities. Whereas Willow is a general purpose aggregation service, DASIS employs aggregates to load balance the underlying peer-to-peer graph. In contrast, DECA is a multilayered hierarchical model which offers an efficient dissemination infrastructure; it organizes participants in a hierarchy of self-contained clusters which are, in practice, locality-aware overlays.

GAP[13] is characterized by the construction and maintenance of a BFS tree on top of the network and consequently, it suffers from the same aforementioned drawbacks.

The most similar abstraction in spirit to ours is Astrolabe[14]. In short, Astrolabe is a distributed management service designed to monitor and report continuously the state of a collection of dynamically changing resources to users. To do it, it organizes the resources into a hierarchy of domains, called *zones*, and associates attributes with each zone. A *zone name*, which is unique and describes its position in the hierarchy, is given to each zone to globally identify it. Similar to us, Astrolabe uses aggregation and a gossip protocol for quickly spreading changes throughout the system. However, it presents three important shortcomings. It is not self-organized since the hierarchy is implicitly defined when the administrators name the zones. It is semi-decentralized as each zone elects a set of hosts, called *representatives*, to gossip on behalf of the zone and it is not fault-tolerant. With only one representative per zone, Astrolabe is highly sensitive to host crashes.

3 A Hierarchical Management Framework

To meet the scalability demands for distributed AGGREGATION, we propose a hybrid technique that blends together the scalability of trees with the resilience of epidemic algorithms.

3.1 Hierarchical Model

We are considering a dynamic P2P overlay graph $G(V(t), E(t))$, where $V(t)$ is the set of vertices at time t, and $E(t) \subseteq V(t)\times V(t)$ is the set of edges that may change over time. Each node x has an associated value $w_x(t)$ at time t. It represents the value that is being subjected to the AGGREGATION function F.

Because G is a DHT, it has a finite m-bit identifier space of 2^m elements denoted as I. Besides, G is also a hierarchical substrate recursively built up from a *proper nesting* of clusters. By a proper nesting, we mean that for any pair of clusters in G, the two clusters are either disjoint, or one is a proper subset of the other. Technically, the latter means that our architecture defines a partial-order tree; nodes are organized into clusters, clusters are organized into superclusters, superclusters are organized into hyperclusters etc. As a result of this partial ordering, the set I is at tier-0, the highest tier, whilst in any subsequent tier-k provided $k > 0$, the potential number of identifiers for each cluster falls off. This occurs because each cluster $C \in G$ is a proper subset of I, that is, $C \subset I$. In other words, there are not duplicate identifiers since $I \supseteq G$ and $C \in G$. To conclude, let L denote the number of tiers.

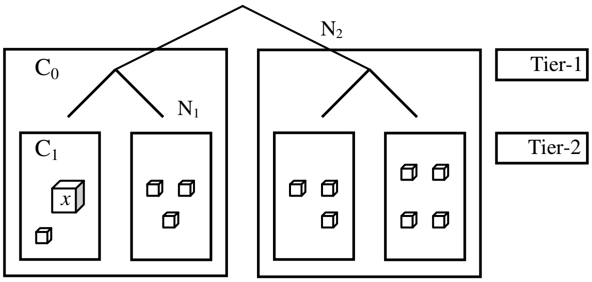


Fig. 1. A DECA hierarchy

Node status. Each node x (that is a peer) is contained in a sequence of telescoping clusters: $C_{l-1}(x) \subset C_{l-2}(x) \subset \ldots \subset C_0(x)$, for some $l \in \{0, \ldots, L-1\}$ and where $C_{l-1}(x)$ denotes the *x's* leaf cluster and $l-1$ its tier depth. For each $C \in G$, we say that C is a leaf cluster if and only if C is not a union of a finite number of other sets in G. In other words, any node is part of a raising sequence of larger groups up to the root. To retain the routing capabilities of the flat design, each node x requests for some routing information at each tier-k, for $k > 0$. Particularly, it is $\acute{H}$ responsibility to provide an efficient hierarchical substrate equivalent in degree and diameter to the flat design. Later in section 4, we discuss in more detail the $\acute{H}$ involvements.

Let $N_i(x)$ be the set of tier-i subtrees not including node x. Let N be the collection of sets $N_i(x)$ for all $i \in \{0, 1, \ldots, L - 1\}$. Then, for each subtree $S_j(x) \in \{N_i(x) \mid \forall N_i(x) \in N\}$ node x knows at least one node, namely, x' in $S_j(x)$. Such a node x' is said to be the x's delegate node in $S_j(x)$. In fact, a delegate node x' can be viewed as a kind of bridge since it lets a pair of nodes exchange its weights $w(t)$ at a given time t. Because function $\acute{H}$ strips G into smaller graphs of the same "*family*", each node x has at least one delegate in each height-k subtree, $k > 0$. The latter is advantageous in that nodes joining or leaving require only local changes in the network. Besides, it isolates faults; it enables effective bandwidth utilization, adaptation to the underlying network and a scalable network management. We note that no global information about the structure of the hierarchy is necessary; it suffices for each node to know its current position in the hierarchy and the list of ancestor groups up to the root. To ease the membership management, we assume that each cluster has a unique group identifier.

When a node x joins the system, x asks an arbitrary existing node, say y, to determine the closest node to x — using a topological aware mapping. Denote this closest node by z. Node x then initializes its routing table with z's routing table. Let *x's* routing table be defined as follows. Let D(x) be the set of x's delegate nodes. Let $L(x)$ be the x's routing table for its leaf cluster. Then, the x's routing table consists of $D(x) + L(x)$ nodes. For robustness to node failures, a "diversity" property should be maintained across the routing tables of nodes in the system. To accomplish this goal, it suffices for each node x to apply the maintenance rules of the flat design along the sequence of telescoping clusters: $C_{l-1}(x) \subset C_{l-2}(x) \subset \ldots \subset C_0(x)$. Therefore, maintenance of the hierarchical network is relatively simple and almost identical to the flat overlay.

– **Network Awareness.** As mentioned above, one of the most challenging questions facing DHTs is whether they can compute aggregates for real-time management. To do it, a common technique consists of integrating the so-called "*proximity*" into the target DHT. In DECA, this means that nodes that are topologically close are organized into clusters; topologically-close clusters are gathered into *superclusters* etc. Because the cluster *nestings* come directly up from the underlying topology, DECA can assume an asynchronous distributed model i.e., a known upper bounds on message transmission delays, and clock drift rates. Hence, we can integrate all the above bounds in a known universal maximum delay Δ between any pair of nodes. A message sent a time t will be received by the destination node within time t to time $t + \Delta$. Also, we can assume that the communication time between two nodes within its leaf cluster is smaller than Δ. This method may introduce errors but exact synchronization is not necessary, so gossiping fully suffices.

3.2 Node Aggregation

In this section, we address the fundamental question regarding DECA: "*What is the actual protocol used to calculate the global function F over data residing at the nodes of a DHT?*" We answer this question proposing a hybrid protocol that combines a gossip-based dissemination mechanism with a tree-based propagation structure that reflects the hierarchical organization of groups. First, a node "*gossips*", within its leaf cluster, about the individual weights it knows about. Then, it computes an estimate of F in a bottom-up fashion using the estimates it receives from *delegate nodes*. As G is expected to materialize into a *proper nesting* of clusters, our algorithm requires about $O(h)$ phases to compute F, where h is the height of the tree.

Protocol overview. The major aim of DECA is to cope with the inaccuracies found in aggregates and provoked by the transient nature of P2P networks, where a significant fraction of nodes become inactive within a short period of time. In such settings, a pure tree-based approach tends to be rather unpractical since a single node failure can cause the root to miss the information of the whole subtree below the fault. In order to achieve more fault-tolerance, we need more redundancy in messages sent. Thus, gossip protocols, which are simple and fault-tolerant — though, at the cost of a higher number of messages —, constitute an attractive alternative. Fortunately, we believe that our scheme far affords the expenses of carrying out such form of massive data dissemination. Recall that clusters are expected to be small and topologic-centric. So scalability is achieved through the distribution of AGGREGATION across the clusters.

Regarding the flow of information, we distinguish between *push*, *pull* and *push-pull gossip protocols*. Assume a node x calls node y. Then, we have:

i. In *push* gossip, the rumor is *pushed* if x tells y the rumor.
ii. In *pull* gossip, the rumor is *pulled* if x requests y for the rumor.
iii. In *push-pull* gossip, the rumor is both *pulled* and *pushed*.

These protocols are reliable in a probabilistic sense. Karp et al. [15] show that if a *push-pull* gossiping is run for $O(\ln n)$ rounds, then, *w.h.p.*, all nodes have the rumor, and in addition, the total number of messages sent is $O(n \ln \ln n)$.

In order to bound the communication cost, we assume a *push* gossip algorithm that can be described as follows. Each node x receives rumors for $O(\log n)$ *epochs*, where an epoch is a fixed time interval of length Δ (network delay for clusters is $O(\Delta)$). In each epoch, each node x gossips to δ other nodes, randomly chosen from the subset of nodes x knows about. We will refer to this subset as x's local view Γ_x. Views may be partial and inconsistent but large enough to ensure a fast convergence.

In *push* gossip, nodes gossip at a constant rate in each round, and therefore, the number of messages sent is $O(n \log n)$. However, we can reduce such overhead by tailoring the choice of targets to the underlying graph topology G, but a more general strategy applicable to any P2P topology is desirable.

To conclude this section, we describe how a node x obtains the estimate for level i, for $(i > 0)$. We call this method GET_ESTIMATE(i, F). Let $S_i(x)$ be the set of all height-i subtrees not including x. Let $\psi_{i,j}(x)$ be the estimate corresponding to subtree-j at height-i not including x. Then, procedure GET_ESTIMATE(i, F) consists of two steps. First, it obtains the estimates $\psi_{i,j}(x)$ from all x's sibling subtrees $S \in S_i(x)$ through the corresponding delegate nodes. Finally, it applies function F over these estimates.

After these preliminaries, we are ready to describe how our algorithm manages to compute F.

Aggregation algorithm. DECA algorithm starts "*simultaneously*" at each node. The algorithm consists of 3 phases:

1. *Phase* 1: this phase lasts $O(\log max\{|C_k|\}_{k \geq 0})$ epochs, where $|C_k|$ denotes the cardinality of leaf cluster k. In order to estimate $O(\log max\{|C_k|\}_{k \geq 0})$, it suffices for nodes to use function MAX over the cardinalities of leaf clusters. In this phase, each node x "*gossips*" about the individual weights stored across its leaf cluster, which of course include x's local weight. To do it, node x *i*) *periodically* (once in every epoch) selects a random subset of nodes from its local view Γ_x and *ii*) sends them an arbitrary weight chosen uniformly at random along with the identifier of the node that keeps it. In turn, x discovers the weights of other nodes of its own cluster the first time it receives them via a *push* message. After $O(\log max\{|C_k|\}_{k \geq 0})$ epochs, x applies the function F to all weights it has collected in its cluster, and bumps itself to phase 2.
2. *Phase* i ($1 < i \leq h + 1$): in phase i, each node x invokes GET_ESTIMATE(i, F) to obtain the estimate for tier-i. Then, x bumps itself to phase $i + 1$. Note that any node has not available the estimate for its height-i subtree until phase i terminates.
3. *Final phase*: when a node x finds itself in phase $i = h + 1$, it has an estimate of the global function F evaluated over the entire DHT. Then, the protocol finishes at x.

Time Complexity. The number of phases for the algorithm is $h + 1$, where h is the height of the hierarchy. Besides, let λ be the set of leaf clusters. Since phase 1 lasts $O(\log c)$ rounds, where $c = max\{|C|: C \in \lambda\}$, the time complexity for the algorithm is $O(\log c + h)$.

Message Complexity. The communication cost is $O(bn \log n + bn)$, where b is the size in bits of the weights and n the number of nodes in the network.

4 The Hierarchical Function ℋ

In general, the problem of finding a suitable ℋ is complex. In [16], Ganesan et. al. provide a framework to transform a variety of DHTs into their hierarchical versions. As a part of our ongoing research, we use instead Cayley graphs, a common technique in design of interconnection networks, to devise the exact ℋ for a given input overlay. Briefly, Cayley graphs are extremely helpful in the analysis of static topologies with regards to quality measures such as diameter or degree — our hierarchical DHTs must preserve the same degree and routing performance as the flat designs. In particular, many Cayley graphs, such as hypercubes, are hierarchical. Further, if an algebraic theoretic model that can elucidate the hierarchical structure of a graph exits, then to procure it a proper ℋ is quite simple: *upon an overlay is proven to be a Cayley graph, it suffices to use the standard definition of hierarchical Cayley graph to approximate it.* Although many Cayley graphs are hierarchical, not all of them admit a recursive decomposition into smaller graphs of the same family. More specifically, we are only interested in those DHTs that are recursively built up by adding isomorphic copies of smaller *Cayley* graphs. A hierarchical DHT is not just a graph, but rather a family of graphs G_0, G_1, G_2 etc. defined for any of the potential static sizes.

Definition 1. Let $\Gamma(V, \circ)$ be a finite group, 1 its neutral element, and let $S \subseteq V - \{1\}$ be closed under inversion (i.e. $s^{-1} \in S$ for all $s \in S$). The Cayley graph $G(\Gamma, S) = (V, E)$ of $(V, \circ)$ is the graph on V where x, y are adjacent if and only if $x \circ y^{-1} \in S$. In other words, there is an edge (x, y) if and only if there exists a generator $s \in S$ such that $x \circ s = y$.

Cayley graphs include a large number of families of graphs, like hypercubes, star and pancake graphs [17] etc. The next definition constitutes the core of our scheme to build up hierarchies from the ground:

Definition 2. Let $< \{s_1,s_2,\ldots,s_i\} >_\Gamma$ be the subgroup of $\Gamma(V, \circ)$ generated by the set $\{s_1,s_2,\ldots,s_i\} \subseteq S$ i.e. the smallest subgroup of Γ which contains $\{s_1,s_2,\ldots,s_i\}$. Let $\Gamma(V, \circ)$ be a finite group and $S \subseteq V - \{1\}$ such that $S = S^{-1}$. Then, the Cayley graph $G(\Gamma,S)$ is said to be hierarchical if there exits an ordering $\{s_1, \ldots, s_k\}$ of the generators of G such that the subgroups $< \{s_1,s_2,\ldots,s_i\} >_\Gamma$ are all distinct.

The above definition yields a surprising outcome. If we order the generators such that each s_{i+1} is outside the subgroup generated by the subgroup $<\{s_1, s_2,...,s_i\}>_\Gamma$, then we can obtain an accurate approximation of ʹH. Specifically, each s_{i+1} incorporates the additional edges required to interconnect the exact number of copies of graph G_i to produce the next family graph G_{i+1}. Therefore, Cayley graphs give us a useful hint to the question of how to get a suitable method to obtain hierarchical substrates from flat topologies. Although Cayley graphs are very helpful, they only give a static solution to the problem of hierarchical construction. In fact, adopting a concrete instance of ʹH involves a further analysis to determine if ʹH properly fits in a dynamic environment where the nodes join or leave independently. However, this topic is beyond the scope of this work.

To conclude, we want to underlie that in our case study, the hierarchical version of Chord we obtain via ʹH preserves the logarithmic bound $O(\log N)$, both in degree and number of routing hops.

5 Whirl: The Hierarchical Version of Chord

In this section, we concisely present the hierarchical version for Chord. First, we give the Cayley graph definition for Chord. Second, we discuss the specific function ʹH which manages to map Chord to *Whirl*, our hierarchical version of Chord. Since a detailed description of Whirl was provided in [5], we omit the irrelevant details from network management viewpoint.

Let Γ be the cyclic group $(\mathbb{Z}_{2^m}, +)$ of 2^m elements with generators 2^i for $i=0\ldots m-1$. The Cayley Graph $G(\Gamma, \{\pm 2^i: i \in \{0,\ldots,m-1\}\})$ is the Chord graph with diameter $m/2$ and degree $2m$.

The next theorem claims that Chord is a hierarchical Cayley graph. It also sets the order for the generators $\pm 2^i$, for $i=0\ldots m-1$, which reveals the in-built ʹH for Chord:

Theorem 1. Let Γ be the cyclic group $(\mathbb{Z}_{2^m}, +)$ of 2^m elements. Let $G(\Gamma, \{\pm 2^i: i \in \{0,\ldots,m-1\}\})$ be the Cayley graph of Chord. Then, Chord is a hierarchical overlay if and only if its generators are ordered as follows: $\pm 2^{m-1}, \pm 2^{m-2}, \ldots, \pm 1$.

Proof. Omitted due to space constraints.

The above definition theorizes that a Chord graph of 2^m elements can be viewed as two interleaved copies of a Chord graph of 2^{m-1} nodes with the additional connections linking adjacent vertices. For example, consider the cyclic group ($\mathbb{Z}_8$, +) for a Chord graph of 8 vertices {0, 1, ..., 7} and degree 6. The ordered set of generators for the *Cayley* graph is {±4, ±2, ±1}. Then, the Chord graph of 8 vertices can be obtained by applying the generator ±1 to two copies of 4 vertices, one with vertex set {0, 2, 4, 6}, the other with vertex set {1, 3, 5, 7} and both with generators {±4, ±2}. It is easy to notice that the vertex sets of both copies are left cosets of group ($\mathbb{Z}_8$, +). Concretely, this leads to a more general implication. It signals that the union of two copies of the graph G_i to produce graph G_{i+1} entails to establish only one additional link per node. Technically, the latter is the reason why Whirl is optimal regarding both degree and diameter. For further details, refer to [5].

Bearing in mind the above facts, function ′H for Chord is two-fold: (*i*) *it retains the standard Chord's rule for link creation* and (*ii*) *performs a simple operation on node identifiers*. The reason to maintain Chord's rule is for enabling "*hierarchical*" lookups in Whirl. In general, a hierarchical lookup works as follows. Suppose a node q looks for an item k. First, q tries to find k within its leaf cluster to take advantage of network proximity. If q finds k the search stops. Otherwise, the query reaches the closest predecessor p of k at this tier. Then, node p switches to the next higher cluster and continues routing on that cluster. By repeating the latter, item k finally is found.

Besides, ′H divides the node identifiers in two parts. A PREFIX of $m - p$ bits and a SUFFIX of p bits provided $0 \leq p \leq m$, where m is the length of node identifiers within Chord circular identifier space $[0, 2^m)$. The SUFFIX determines the cluster of a node (*cluster* ID) whereas the PREFIX, drawn uniformly at random, specifies the identity of a node inside the actual cluster (*node* ID). Then, the application of the generator ±1 to any pair of neighboring clusters with an m-bit identifier space, produces a larger cluster with a (m+1)-bit identifier space and "*ring*" links between adjacent vertices. Also, the generator ±1 installs the delegate nodes at that tier. As a result, merging a pair of clusters of the same level is achieved easily by *i*) *contracting the SUFFIXes* and *ii*) *extending the PREFIXes* of nodes one bit, respectively. The last result has a broader scope: it lets nodes reuse the links of "lower" clusters for routing at "higher" tiers. As simulation results demonstrate in next section, our theoretic model is capable to elucidate a hierarchical *degree-optimal* version of Chord.

6 Simulation Results

We now present a battery of tests to evaluate the functionality and performance of our framework through simulation. Because the underlying substrate is a Chord overlay, maintenance costs are of the same order as Chord. The main hypotheses we want to evaluate are:

1. *The basics of Whirl*: the degree distribution and average number of routing hops to demonstrate the quality of function ′H.
2. *The proximity*: the capacity that offers a hierarchical DHT for adaptation to the underlying network in comparison to its flat design.
3. *The performance of DECA*: it includes the convergence time for a tailored *push* gossip algorithm, applicable to Chord; the communication cost in terms of number of messages sent and the accuracy of our solution.

All of the experiments for this section use a hierarchy with a fan-out of 2 at each internal node. The number of tiers in the hierarchy varies from 1, plain Chord, to 5.

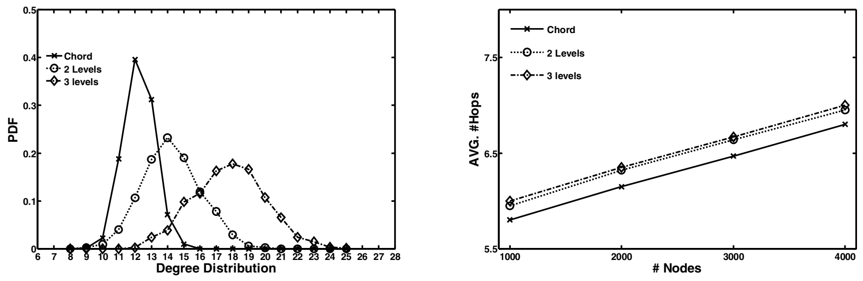


Fig. 2. a) Number of links distribution **b)** Average Number of Routing Hops

The number of nodes oscillates from 1K to 4K, and all the nodes choose a random 14-bit identifier. Nodes are evenly distributed across the leaf groups.

$\mathcal{H}$ Test. Our first set of experiments evaluates function $\mathcal{H}$ abilities. In particular, the *distribution of the number of links* and *the average number of routing hops* are provided. Regarding the first metric, figure 2.a) plots the distribution of number of links for a 1K-node network in a 1-level (Chord), 2-level and 3-level hierarchy. We observe that for Chord this distribution is peaked around the average of 12 links/node. As the number of tiers in the hierarchy increases, the mean shifts to the right whilst the distribution "*flattens out*" to the left of this value. Although the number of links is not exactly the $\log|V|$, we see that the average number of links is $\log|V| + c$, i.e. $O(\log|V|)$, where c is a small constant that depends on the number of tiers. On the other hand, figure 2.b) depicts the average number of hops required to route between two nodes as function of the network size. We see that the number of routing hops is extremely close to $\log|V|$ irrespective of the number of tiers. To evaluate the above metric, we inserted 1K items into the system. Later, ten nodes were requested to retrieve all items. For each item, a "*hierarchical lookup*" was issued and the number of hops it spent, accounted. In conclusion, DECA through function $\mathcal{H}$ is capable to produce hierarchical versions of flat DHTs that preserve the same degree and number of routing hops.

Proximity Test. In this test, we evaluate DECA routing performance in terms of network delay. To make a fair comparison, both the flat Chord and its hierarchical version have the same number of nodes, denoted $|V|$. In order to measure DECA adaptation to the physical network, we use GT-ITM[18] topology generator to produce a 100-node highly connected *backbone*. For each node in the *backbone*, we attach a number of graphs representing *stub* domains. The latency weights are: 10ms for *backbone* edges, 100ms for *backbone-stub* edges and 5ms for *stub-stub* links. To construct the desired $|V|$-node network, we attach each node to a *stub* through a 1ms-latency edge. Figure 3 depicts the relative performance of an arbitrary 3-level, 4-level Whirl graph versus a plain Chord without proximity. Clearly, it illustrates that Whirl widely outperforms Chord.

Performance Test. This test evaluates DECA performance in terms of *convergence time*, *number of messages* sent and degree of *accuracy*. To this end, we implement a *push* gossip protocol over Chord partially based on the foundations of *lbpcast* [19]. Each node x maintains a partial view Γ_x of the system that varies as node x discovers other nodes via *push* messages. Also, we employ COUNT function for our analysis. Basically, this test aims to count the total number of nodes in the overlay. For all the

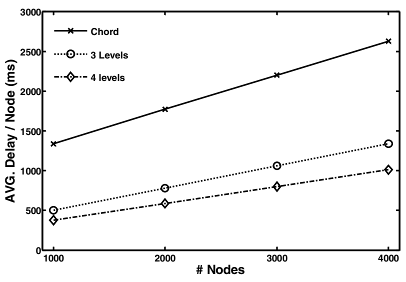


Fig. 3. Average delay between nodes

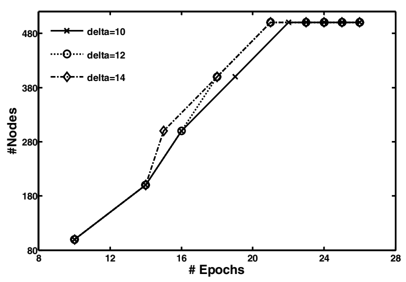
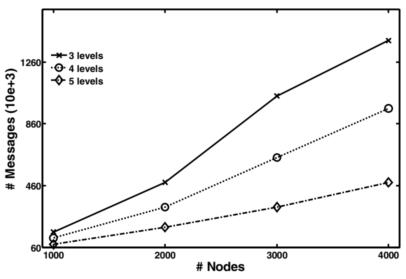


Fig. 4. Convergence Time

tests, the probability of a node crash (without recovery) in every epoch and in every phase i (i >0) is $pf = 0.001$.

Figure 4 shows the relationship between the "*gossip*" rate δ and the number of rounds that it takes to aggregate an event in a leaf cluster. The figure shows that increasing δ decreases the number of necessary *epochs* to aggregate a value, but conveys also the fact that the gain is not proportional. Figure 5.a) illustrates the cost in number of messages that DECA spends. Note that leaf clusters are expected to be several orders of magnitude smaller than a flat DHT. Then, it is easy to see that as the average depth of the hierarchy increases, the overhead sharply reduces. Finally, figure 5.b) presents the fraction of nodes whose estimation of COUNT function is outside the range [0.9|V|, 1.1|V|]. The high degree of *accuracy* comes from: *i*) the high degree of reliability provided by the *gossip* protocol and *ii*) from the small quantity of delegate nodes required to aggregate. The advantage of using delegates is that the maintenance algorithm of the flat design carries out their refreshing.



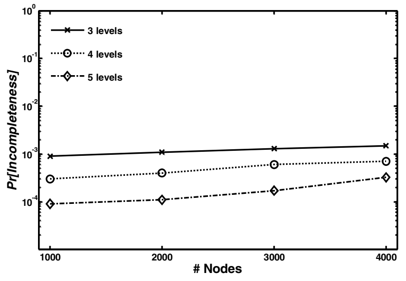


Fig. 5. a) Number of messages sent (10e+3) **b)** Incompleteness probability

7 Conclusions

In this paper, we have expressed the need for a scalable AGGREGATION framework for DHTs as a basis for wide-area management architectures. We have argued the reason why traditional approaches for solving this problem do not scale in large groups, and do not perform well over fault-prone networks. To cope with this, we have sought for new theories that fit neatly into Peer-to-Peer paradigm foundations. In this line, we have introduced a hierarchical abstraction that manages to extend AGGREGATION to P2P wide-area networks. Also, a hierarchical methodology based on Cayley Graphs, which produces hierarchical systems from the usual *flat* DHTs, has been unveiled. Finally, an AGGREGATION algorithm which is hybrid, since it combines the fault-resilience of gossip algorithms with the scalability of trees, has been carefully devised to achieve the scalability/fault-tolerance requirements of large-scale P2P networking.

References

1. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet application". In ACM SIGCOMM, 2001.
2. S. Ratsanamy, P. Francis, J. M. Hellerstein, and S. Shenker. A scalable content-addressable network. In ACM SIGCOMM, 2001
3. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In IFIP/ACM Middleware, 2001.
4. B. Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. IEEE Journal on Selected Areas in Communication, 22, 2004.
5. M.S. Artigas, P. García, J. Pujol, A. F. Skarmeta. Cyclone: A novel design schema for Hierarchical DHTs. In Proc. 5th Conf. On P2P Computing, 2005.
6. S. Madden, M. J. Franklin , J.M. Hellerstein, W.Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. SIGOPS Operating System Review 36 (2002).
7. M. Bawa, H. García-Molina, A. Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. In *Manuscript*, 2003.
8. Ji Li, Karen Sollins, Dah-Yoh Lim. Implementing aggregation broadcast over distributed hash tables. In ACM SIGCOMM, 2005.
9. G.Doyen, E.Nataf, and O.Festor. A hierarchical architecture for a distributed management of P2P networks and services. In Proc. of DSOM'05, 2005.
10. Z. Zhang, S.-M. Shi, and J. Zhu, SOMO: Self-Organized Metadata Overlay for resource management in p2p DHT. In Proc. IPTPS'03, Feb. 2003.
11. R. Van Renesse, and A. Bozdog. Willow: DHT, Aggregation, Publish/Subscribe in One Protocol. In Proc. IPTPS'04, Feb. 2004.
12. K. Albrecht, R. Arnold, and R. Wattenhofer. Aggregating Information in Peer-to-Peer Systems for Improved Join and Leave. In Proc. 4th Conf. On P2P Computing, 2004.
13. M. Dam and R. Stadler. A Generic Protocol for Network State Aggregation. In Proc. Radiovetenskap och Kommunikation (RVK), 2005.
14. R. Van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. ACM Trans. Comput. Syst., 21(2), 2003.
15. R. Karp, C. Schindelauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. In Proc. 41st IEEE Symp. on Foundations of Computer Science, 2000.
16. P. Ganesan, G. Krishna, H. García-Molina. Canon in G major: Designing DHTs with hierarchical structure. In Proc. ICDCS, 2004.
17. Akers, S.B., Krishnamurity, B.: A group-theoretic model for symmetric interconnection networks. IEEE Trans. Comput. Vol. 38, 1989.
18. E.Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In Proc. INFOCOM96, 1996.
19. P. Eugster, R. Guerraoui, S.B. Handurukande, A.-M Kemarrec, P.Koutnetsov. Lightweight probabilistic broadcast. In Proc. Conf. Dependable Systems and Networks (DSN), 2001.

Towards Distributed Hash Tables (De)Composition in Ambient Networks

Lawrence Cheng[1], Roel Ocampo[1], Kerry Jean[1], Alex Galis[1], Casba Simon[2], Robert Szabo[2], Peter Kersch[2], and Raffaele Giaffreda[3]

[1] University College London, Electrical Engineering Department, Torrington Place, London, WC1E 7JE, UK
{l.cheng, r.ocampo, k.jean, a.galis}ee.ucl.ac.uk
[2] Budapest University of Technology and Economics, Dept. of Telecoms and Media Budapest, Hungary
{simon, szabo, kersch}@tmit.bme.hu
[3] British Telecommunications PLC
raffaele.giaffreda@bt.com

Abstract. When different wireless networks come in close proximity there is often a need for them to logically combine, or *compose*. We focus on a known research problem particularly in Ambient Networks (ANs), where heterogeneous Distributed Hash Tables (DHTs) contained in these wireless networks need to merge or divide as a result of these dynamic (de)composition processes, respectively. We present two novel DHT (de)composition models for ANs, known as *absorption* and *gatewaying*, that are designed to handle (de)composition of DHTs in different AN network environments, with minimal disturbance to existing member nodes.

Keywords: Ambient Networks, Composition, Decomposition, Distributed Hash Tables.

1 Introduction

The aim of the Ambient Networks (ANs) project [7] is to develop the next generation wireless networks. An AN consists of potentially large numbers of independent, heterogeneous mobile nodes that can logically interact with each other to share a common control space, known as the Ambient Control Space (ACS) [7], for resource sharing. Given that an AN may consist of large number of AN nodes, there is clearly a need for a distributed and scalable management data storage and retrieval mechanism for each AN. Previous research works [1][2][4] have suggested that Distributed Hash Tables (DHTs) is a candidate. Example DHTs are Content Addressable Network (CAN) [3], Chord [6], Pastry [5], and others. However, much of the existing research mainly focuses on optimising DHT routing and scalability; and usually assumes a common DHT across the entire network that any nodes can join [2]. Since ANs may constantly compose and decompose with other ANs[1], we argue

[1] By AN composition, we refer to process of which the ACSs of two (or more) ANs interact with each other, to establish a common ACS between the two (or more) ANs for resource sharing. AN decomposition refers to the process of a common ACS being divided [7].

that the successful use of DHTs in implementing various distributed management components in ANs depends on an ability to efficiently *compose* and *decompose* DHTs. By DHT (de)composition, we refer to member nodes of homogeneous or heterogeneous[2] DHTs of different ANs interacting with each other to share distributed information.

The challenge is that DHT (de)composition in ANs must be conducted in a resource-limited environment i.e. a wireless environment. Thus, in addition to the need for a more efficient underlying routing algorithm in DHTs (which is beyond the scope of this paper), there is a need to minimise the disturbance caused by the (de)composition process to existing member nodes of the DHTs. By minimising the disturbance, we mean to minimise: (a) the amount of network overhead incurred by the (de)composition process, and (b) the amount of storage data that needs to be (re)distributed to other nodes in the DHT *after* (de)composition has completed (during which keyspace might have been re-assigned to new members). As far as we are aware, there has not been a vast amount of research work on DHT (de)composition; a DHT merging process designed for DHTs based on the Chord protocol was presented in [2] (see later). In this paper, we shall outline two novel (de)composition models for CAN-based DHTs for AN. We shall also discuss the mappings of our models to DHT implementations other than CAN. We start our investigation based on CAN-based DHTs because of its design simplicity, which could help readers understand this new research challenge (of DHT (de)composition). Furthermore, this would enable us to gain a better understanding of the design requirements of DHT (de)composition, and put ourselves in a better position to evaluate, experiment with, and to tailor a generic DHT (de)composition model that would also cover other DHTs.

2 Background

In this section, we will start off with explaining some of the assumptions that we have made when designing our approaches. Then, we shall provide an overview of our approaches, by discussing their unique features and triggering factors.

2.1 Assumptions

To simplify our discussion, we assume that each AN has DHT-based management components. Each AN node will have its own keyspace in the DHT once it has joint a DHT; and each node maintains a coordinate routing table that keeps IP addresses and virtual coordinate zones of its neighbours in the approach as indicated in [3]. To simplify our discussion, we assume one DHT establishment per AN. Remember that we are interested in DHT (de)composition in this paper; thus, (pre)establishment of DHT in each AN is assumed. Readers are referred to [1][8] for more detail on optimized DHT establishment techniques for wireless networks.

[2] By homogenous DHTs, we refer to DHTs that use keyspace of the *same* keysize (e.g. both uses 160-bit keyspace). By heterogeneous, we refer to the opposite (e.g. 160-bit keyspace Vs. 256-bit keyspace).

2.2 An Overview on the Two Models

We present two DHT composition models for ANs, known as *absorption* and *gatewaying*. The absorption model (Fig. 1a) refers to two (or more) individual DHTs (that are owned by two or more ANs respectively) completely merging together, resulting in one *uniform* DHT across the composing ANs. The gatewaying model (Fig. 1b) refers to *bridging* two (or more) individual DHTs together without modifying their original keyspace. Both approaches enable information sharing between DHTs, but are tailor-designed to accommodate *different* network environments (see later).

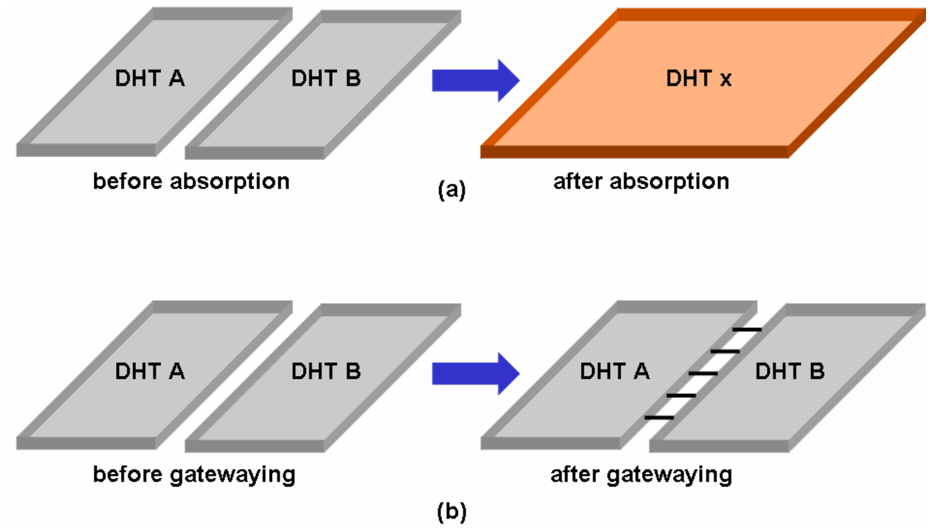


Fig. 1. The generic models of absorption and gatewaying

2.3 Triggering Factors for the Two Models

Although we do not intend to specify the exact criteria when absorption or gatewaying between DHT should be triggered[3], but to illustrate the differences between absorption and gatewaying, first, consider this deployment scenario: when a coach arrives at a train station, an unknown (but potentially large) number of passengers will be getting off the coach and walking towards the station. Assume each passenger is a passenger node, and DHTs have already been established among member nodes of the station and the coach respectively. Because the number of passenger nodes getting off the coach is unknown (i.e. a very dynamic situation), it would be difficult to establish a *fresh*, *new* DHT (e.g. passenger DHT) among members of such a highly dynamic group in *real-time*. Instead, because the station DHT is readily available, should

[3] We believe these issues should be defined by the corresponding AN/DHT Service Providers (SPs) (readers are referred to [7] for more details). Also, SPs should define policies for guiding nodes when nodes are presented with multiple joining offers from different (nearby) DHTs.

passenger nodes (those getting off the coach) wish to become members of a DHT (say, to share information), they should be *absorbed* into the (more static) DHT (i.e. the station DHT). The absorption model is therefore designed to minimise the network overhead on members (whether they are existing members of a DHT or not) when many nodes attempt to join practically at the same time.

The gatewaying model is suitable when one (or more) of the ANs/DHTs come together within reachable distance, but member nodes of the AN/DHTs remain (relatively) static. For example, when two wagons are joined together at an intermediate train station, member nodes of the two wagons are likely to be static (some passengers might get off the train but the majority would stay). There is a need to bridge between the DHTs, so that the DHTs can share information.

3 The Absorption Model

3.1 The Protocol

One way of enabling absorption (as identified in [2]) would be to allow nodes (of a discarded DHT, or do not belong to any DHT) to join a stable DHT, by negotiating with nodes of the stable DHT individually using the standard procedure as [3]. Typically this would require each of the joining nodes to randomly select a keyspace, and obtain keyspace directly from the node that "owns" it in the stable DHT. We refer this as *simple merging*. The advantage of simple merging is its simplicity, i.e. no changes to the existing protocols are needed. But the drawback is that *all* key-value pairs hosted on member nodes of the discarded DHT or on individual nodes would have to be re-distributed to nodes of the stable DHT. Also, if keyspace is randomly selected, many nodes in the stable DHT must also update their neighbourhood information each time a new node joins (see later for more details). In the case of large scale deployment, this could potentially create unnecessary network traffic, which is not desirable, particularly in wireless networks.

We suggest that, absorption negotiations should be conducted through the point(s) of contact between the (two) ANs only. Points of contact are the nodes that have *physical connections* with other ANs. For example, Y2 and Y3 are the points of contact of AN/DHT Y (Fig. 2); whereas X1 and X4 are the points of contact of AN X. In this example, we assume that nodes in AN/DHT Y (i.e. the coach AN/DHT) will join with AN/DHT X (because the station AN/DHT is more stable). Instead of Y2 and Y3 randomly selecting points in the keyspace of any nodes of DHT X (which is the case in [2][3]), X1 and X4 will give up some of their keyspace to Y2 and Y3 respectively, by carefully selecting appropriate keyspace from *within* the keyspace that they own. Note that the entire absorption process is a transient process, which ceases to operate after a timeout. After the timeout, new nodes will be joining the (unified) DHT under the normal procedure; that is, by randomly selecting keyspace from any nodes in the DHT (see later).

Note that a key feature of absorption is that keyspace is selected within a keyspace (instead of splitting), with the goal of reducing the level of disturbance to neighbouring nodes. Fig. 3[4] shows the resultant keyspace partitioning under different

[4] For simplicity, we illustrate our examples using a 2-dimentional coordinate space.

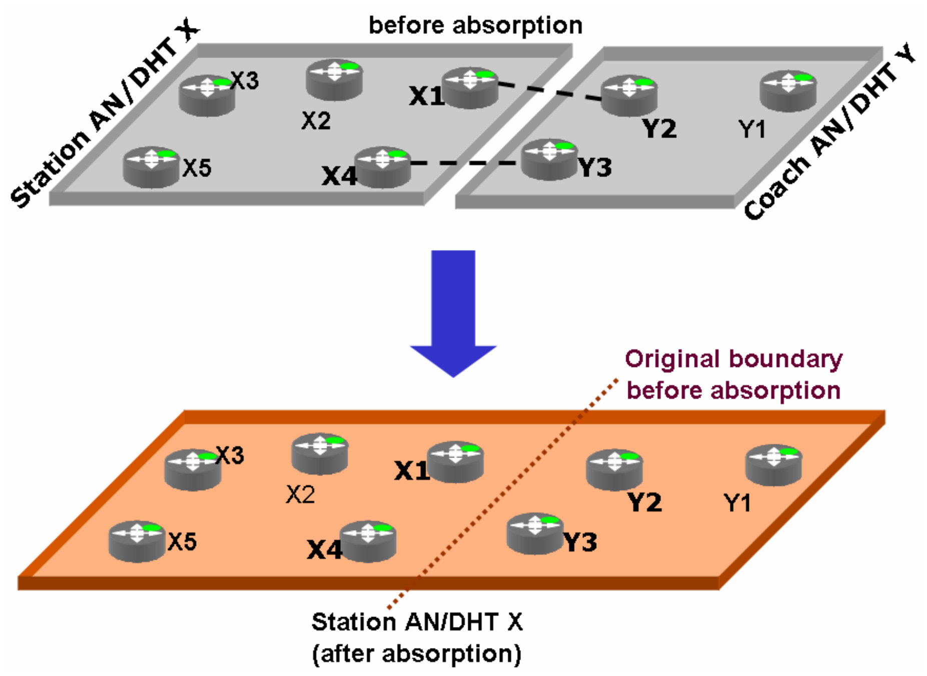

before absorption
Station AN/DHT X
Coach AN/DHT Y
X1
X2
X3
X4
X5
Y1
Y2
Y3
Original boundary before absorption
Station AN/DHT X (after absorption)

Fig. 2. The absorption model

approaches. Fig. 3a shows the original keyspace of DHT X (before composition). If nodes of DHT Y are allowed to randomly select keyspace from member nodes of DHT X, or if X1 and X4 simply split up their keyspace for Y2 and Y3 respectively, the resultant keyspace might end up as shown in Fig. 3b. As a result, X1, X2, X3, X4 and X5 would *all* have to update their neighbourhood information. This is obviously less desirable. However, by carefully selecting keyspace within the owner's keyspace (Fig. 3c), only node X1 and X4 would have to update their keyspace respectively. Once dedicated members of DHT Y (i.e. Y2 and Y3) have been assigned with keyspace, they will (re)distribute the keyspace to other members of DHT Y wishing to join DHT X (i.e. Y1) (Fig. 3d). This arrangement is again to minimise the disturbance to existing member nodes of DHT X.

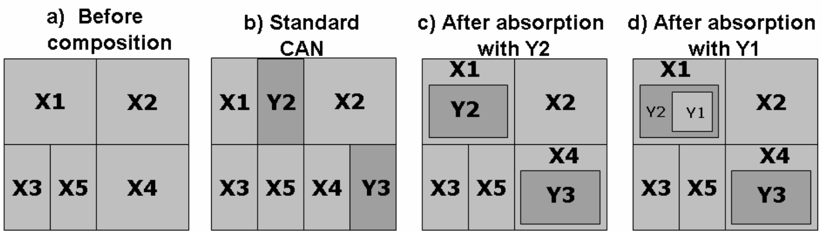


Fig. 3. Resultant keyspace ownership in different approaches

Imagine if all the remaining nodes of DHT Y (i.e. just Y1 in this case, and potentially many more) join DHT X using the standard procedure: if the scale is large (i.e. many nodes joining at once), existing member nodes of DHT X would have to expend significant resources on the tasks of keyspace partitioning and updating neighbourhood information. When absorption is used, remaining nodes (e.g. Y1) should *not* obtain keyspace from nodes other than their points of contact (i.e. Y2 and Y3). When node Y1's `join_DHT` request traverses through Y2 (or Y3), Y2 should terminate the request; Y2 should select *within* the portion of its assigned keyspace, and return the selected keyspace to Y1. The same approach is repeated between Y1 and other nodes of DHT Y: when Y1 has intercepted a `join_DHT` request from other nodes of DHT Y, Y1 will terminate the request, and will response with a selected portion of keyspace within its own keyspace. In this way, each node is responsible for (re)distributing keyspace; thus, a distributed approach for keyspace (re)distribution among the joining nodes is achieved.

3.2 Discussion

The advantage of the absorption approach is that once keyspace has been assigned by X1 and X4 to Y2 and Y3, all other existing member nodes of DHT X (except X1 and X4) are not disturbed. The process is entirely transparent to other nodes that are existing members of the original DHT X; and other nodes that were members of DHT Y may join DHT X through Y2 and Y3 (and subsequently through Y1 once Y1 has obtained its keyspace from Y2 and Y3), which is also a transparent process to existing member nodes of DHT X. Unlike the simple merging approach, our approach requires only a few nodes of the DHT (i.e. DHT X) to be disturbed, and with much less network traffic (i.e. negotiation and communications between AN/DHT are conducted between the points of contact only). Furthermore, because each node is capable of (re)distributing keyspace, keyspace distribution is achieved in a distributed and scalable manner.

It may appear that the requirement of explicit assignation of keyspace by keyspace owners (e.g. Y2) to new joiners (e.g. Y1) violates the random balancing rules in DHTs (i.e. nodes should randomly select keyspace for balance loading). However, we argue that absorption is a transient procedure. After the timeout, new nodes must join through the normal procedure; the keyspace would eventually be randomly and evenly distributed on average. It is possible that all nodes in the DHTs have physical connectivity with each other (i.e. a fully meshed structure of physical connectivity). In this case, one may argue the use of absorption, because effectively all nodes are points of contact. However, if all (wireless) nodes are physically interconnected, then the nodes must be within close physical range. This implies that the number of participating nodes in this merging would be limited. Thus, the amount of network traffic created by, say, a simple merging, would have much less effect. It should be noted that the use of dedicated points of contact for handling keyspace does not affect scalability of the absorption model. The points of contact are responsible for *initially* collecting a portion of keyspace from nodes of the other DHT (i.e. DHT X), and redistributing keyspace to their *immediate* neighbours only. Once the immediate neighbours have obtained their keyspace from the points of contact, the immediate neighbours shall intercept (and terminate) any traversing `join_DHT` requests from other member nodes (of DHT Y), and (re)distribute keyspace to those nodes. Therefore, keyspace (re)distribution to member nodes of DHT Y is carried out in a distributed fashion.

Once new nodes have joint a DHT, they may distribute their local data to other nodes in the DHT if desired (i.e. the `put(key, value)` operation). To minimise traffic caused by many nodes putting data onto many other nodes at one time, provisioning [4] is made in our approach to upload pointers only, instead of the actual piece of data. For example, if Y3 needs to put a piece of data on X2, a pointer is put instead of the actual piece of data. The pointer refers to the actual location of where the data is residing on (e.g. the data source). Thus, the amount of data storage traffic caused by (many) new joining nodes is reduced. One may argue that this arrangement increases the round-trip delay for retrieving a piece of data. However, this approach is ideal for situation where the data to be stored requires frequent updating, such as real-time bandwidth monitoring data. Instead of the data source continually updating the data values on a remote node, a requester – once obtained a pointer to the data source through the DHT – can contact the data source *directly* (readers are referred to [4] for more details).

The arrangement of giving out keyspace to a physical neighbouring node also enhanced routing locality. This is because the resultant overlay DHT neighbourhood reflects the underlying physical neighbourhood. In the standard CAN approach, keyspace ownership are randomly distributed. This means that two neighbouring overlay nodes may be in fact physically distanced. This has a major impact on routing especially in wireless networks; because if routing locality is not addressed, the overhead to route from one overlay node to another can be significantly much higher. However, at the time of absorption, routing locality is optimised in the portion(s) of the keyspace that is being given out during absorption. Note that this approach does not result in creating de facto gateways. It may appear that in Fig. 2 the absorption approach turns X1 to be the de facto gateway of Y2 on the DHT overlay (because to route to Y2 you must always route through X1). However, according to the underlying network connection, X1 is the physical gateway to Y2 *prior to* absorption beings. Therefore, the absorption approach does not create new gateways, but the overlay gateways are the results of absorption that reflects the underlying physical network. Furthermore, we have discussed that the absorption model is a transient process: so in the longer run, when more nodes join through the standard CAN approach, the overlay routing will become more balanced.

4 The Gatewaying Model

4.1 The Protocol

We have mentioned in an earlier section that through gatewaying, the composed AN/DHTs would be able to share information, but at the same time retaining their original keyspace. Existing DHT approaches usually assume only one common DHT (i.e. one common keysize); however, due to the dynamic and heterogeneous nature of ANs, there is a need to support composition between DHTs of different keysize. Thus, there are two environments in which gatewaying may be deployed: gatewaying between DHTs that use keyspace of the same keysize; and gatewaying between DHTs that use keyspace of different keysize. When gatewaying between DHTs of the same keysize, nodes of the composing DHTs are notified of the existence of the other DHTs that are now becoming accessible, as well as their own gateways to the other DHTs[5]. Gateways are the points of contact which have physical connections with nodes of another DHTs; but they serve a different purpose from the points of contact in the absorption model.

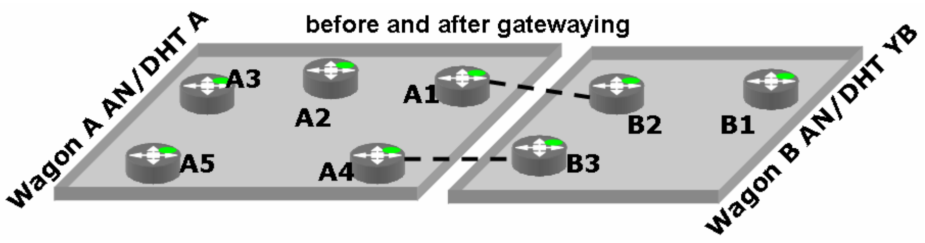


Fig. 4. The gatewaying model between two wagon AN/DHTs

For example, nodes of DHT A (Fig. 4) must be notified that A1 and A4 are the gatewaying nodes to another DHT (i.e. DHT B). The gateways do no more than notification (i.e. they will not request for keyspace). To gain access to another (gatewayed) DHT, member nodes would need to maintain the state of at least one of their gateways[6]. Let's say A5 would like to retrieve a piece of information (after DHT gatewaying). We provide two options for this node to do a search over the gatewayed DHTs: (a) a sequential minimal-evaluation search, and (b) a parallel search. Sequential minimal-evaluation search is ideal for locating unique pieces of data (e.g. a particular video file); whereas parallel search enables a node to locate network-wide information e.g. a node that needs Internet access may wish to search all gatewayed ANs for the best available Internet throughput link that.

In the sequential minimal-evaluation search, A5 computes the location of the information in its home DHT (DHT A) in the same way as stated in the standardised protocol [3], and tries to get the information from the node (of DHT A) that is supposed to hold the information. Suppose the information of interest is not stored in DHT A (i.e. data not found); with gatewaying, the search will continue searching for the same piece of information from other gatewayed DHTs (i.e. DHT B). The node of DHT A which fails to provide A5 with the requested information, say, A3, will inform one of its gateways (i.e. A1), and requests the gateway to search for the same piece of information in other (gatewayed) DHTs. The request is conducted through the gatewaying nodes i.e. A1 and B2. B2 (i.e. the corresponding gatewaying node of DHT B) will try to locate the piece of information on behalf of A1, and will fetch

[5] As an initial approach, notifications are sent to member nodes of a DHT through multicast. Multicast is chosen for its simplicity.

[6] Ideally for robustness, member nodes should maintain as much state of its gateways as possible. However, there is a trade-off between overhead and robustness. As an initial design, we require each node to maintain at least one of its gateways.

over the results to A1 (and subsequently node A5) if the information can be located in DHT B. The search terminates as soon as the information is located (hence the term *minimal evaluation*), and is not forwarded to other DHTs that may be similarly gatewayed in this scenario. In contrast, in a parallel search, as the name implies, the query is forwarded simultaneously (through the gateways) to all DHTs through their respective gateways.

If the composing DHTs' keyspace are of different keysize, we use a similar approach as above, except that B2 must use the correct hash algorithm to compute the correct location in DHT B. For instance, if DHT A's keyspace is 160-bit whereas DHT B's keyspace is 256-bit, B2 must use SHA-256 to compute the correct keyspace of the requested information. Note that when a new AN node wishes to join a DHT that has already been gatewayed, the new node joins the DHT that it is in contact with.

4.2 Discussion

The advantage of gatewaying is its simplicity and reduction in network overhead. It enables information retrieval across DHTs without modifying existing keyspace structure. Thus there is no need to update neighbourhood information on each node in the gatewayed DHTs (which would be required in simple merging or, to some extent, absorption). Also, the chances of successfully retrieving a particular piece of information increase as the number of gatewayed DHTs increases; which enhances the robustness of the overall data retrieval process (i.e. more likely to locate the piece of data of interest). The scale of state maintenance at one gateway is not dependent on the size of the neighbouring networks/DHTs, but depends only on the number of immediate neighbouring gateways, which makes our approach scalable. More importantly, because there is no change to ownership of the keyspace of the gatewayed DHTs, there is no need for data (pointer) (re)distribution, which reduces network overhead. The downside is that the gatewaying model is only applicable when member nodes of the to-be-gatewayed DHTs are relatively static. It may appear that the use of gateways for inter-DHT communications would result in some centralised processing (on the gateways). However, it should be noted that not all `get(key, value)` requests are processed by the gateways; for example, cross-DHT data search takes place only when data retrieval within a DHT fails in sequential minimal-evaluation search, and stops as soon as the data of interest is found; whereas parallel search is used for searching specific-types of information only (e.g. network-wide information). This mode therefore trades off slightly higher traffic costs and processing overhead (on the gatewaying nodes), to enhance overall robustness of the data retrieval process (when comparing to searching for data within one DHT only i.e. no gatewaying), and achieves potentially faster and more comprehensive searches over the entire composed space, with the possibility of returning multiple results from all gatewayed DHTs.

5 DHT Decomposition in ANs

By AN decomposition, an AN is virtually divided into two (or more) ANs, the decomposed ANs do not recognise the existence of each other in the view of control and management[7]. There are several decomposition scenarios. Decomposition between DHTs that were composed through gatewaying is the simplest form. This may happen when, using our previous train scenario for this example, two wagons (which were gatewayed) are detached. Nodes in the gatewayed DHT are informed that the other DHT no longer exists, and the DHTs are said to be decomposed. Decomposition of nodes of a DHT that share one unified keyspace (i.e. may have previously composed through absorption) is slightly more complicated. A node may leave a DHT without establishing its own DHT or joining with another DHT. For instance, a train passenger switches off his laptop. This situation can be considered as a node departure, and can be handled through the standard procedure as specified in [3]: the departing node either handovers its keyspace to a neighbour (i.e. a clean approach), or the unoccupied keyspace will be taken over by its neighbours when its neighbours think it is dead [3]. In other circumstances, a set of nodes may decompose from a DHT, and would like to have a DHT of their own but using the original assigned keyspace. This can be achieved by the departing nodes simply by expanding their own keyspace as if some other nodes have departed from the DHT (Fig. 5). For instance, if 2, 3, and 5 are the departing nodes, they will expand to occupy the remaining space that appears to be "left over" by 1 and 4. The same applies to those who did not leave the DHT. The result would be two separate DHTs, but nodes would be able to retain the original assigned keyspace structure. The last scenario would be when some departing nodes decided to create a new DHT among themselves. In this case, they must discard the original DHT, and creates a new DHT from scratch.

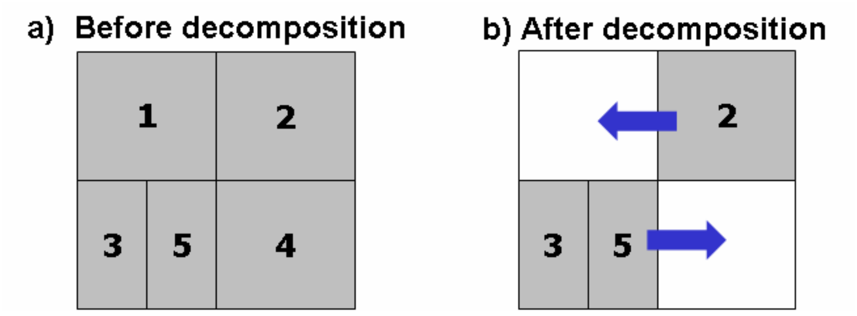


Fig. 5. DHT Decomposition

6 Conclusion and Future Work

We presented two novel models for DHT (de)composition in ANs under different networking environments. We have discussed how our designs minimise the disturbance to existing member nodes during DHT (de)composition, taken into account scalability and efficiency. We believe the research work presented in this paper gives us the insight for developing more generic models for other DHT

[7] This does not necessary mean the decomposed ANs are *physically* disconnected: we are referring to a separation of control space during decomposition of ANs.

implementations such as ring-based DHTs. As future work, we intend to investigate further into the inter-node communications during DHT (de)composition in ANs, and deploying our approaches on other DHT protocols.

Acknowledgement

This paper describes work undertaken in the context of the EU-funded IST Ambient Networks (Phase 2) project.

References

1. H. Pucha, S. Das, Y. Hu, "Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks", in Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), English Lake District, UK, Dec 2004.
2. T. Heer, S. Gotz, S. Rieche, K. Wehrle, "Adapting Distributed Hash Tables for Mobile Ad Hoc Networks", in Proceedings of the 4th IEEE International Conference on Pervasive Computing and Communications Workshop (PERCOM), Pisa, Italy, March, 2006, pp. 173-178.
3. S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A Scalable Content-Addressable Network", in Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM), San Diego, CA, USA, August 2001, pp. 161-172.
4. R. Ocampo, L. Cheng, K. Jean, A. Prieto, A. Galis, "Towards a Context Monitoring System for Ambient Networks", to appear in the Proceedings of the 1st International Conference on Communications and Networking in China (Chinacom), Peking, China, 2006, temporarily available at: http://www.ee.ucl.ac.uk/~lcheng/Papers/CHINACOM_2006.pdf
5. A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", in Proceedings of the IFIP/ACM Middleware, Heidelberg, Germany, pages 329-350, November, 2001, http://research.microsoft.com/ ~antr/ PAST/pastry.pdf
6. I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", in Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM), San Diego, CA, USA, August 2001.
7. R. Campos, C. Pinho, M. Ricardo, J. Ruela, P. Poyhonen, C. Kappler, "Dynamic and Automatic Interworking between Personal Area Networks using Composition", in Proceedings of the 16th IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), Berlin, Germany, Sep 2005.
8. T. Zahn, J. Schiller, "MADPastry: A DHT Substrate for Practicably Sized MANETs", in Proceedings of the 5th Workshop on Applications and Services in Wireless Networks (ASWN), Paris, France, Jun 2005.

CMDB - Yet Another MIB? On Reusing Management Model Concepts in ITIL Configuration Management

Michael Brenner, Markus Garschhammer, Martin Sailer, and Thomas Schaaf

Munich Network Management Team
University of Munich
Oettingenstr. 67
D-80538 Munich, Germany
{brennera, garschha, sailer, schaaf}@mnm-team.org

Abstract. According to ITIL, a CMDB (Configuration Management Database), containing a logical model of the IT infrastructure, forms the basis for effective and efficient IT Service Management. However, a common understanding of what constitutes a CMDB has not yet been established. By contrast, concepts for building and using MIBs (Management Information Base) – also aimed at providing logical models of the IT infrastructure – have long since been established in the area of systems management.

This paper presents an overview of the CMDB and MIB concepts, discusses how they relate to each other and compares them based on the main purposes of a CMDB. It discusses whether modeling approaches used for MIBs can be reused for CMDBs. To this end, a criteria catalog based on core CMDB concepts and basic information requirements of ITIL's Service Management processes are derived, and the challenges of implementing a CMDB reusing concepts of common management models are discussed. Concluding, basic approaches towards integrating CMDBs and MIBs are presented.

1 Introduction

In approaching ITSM *(IT Service Management)* issues, there is a current trend towards greater consideration of organizational (rather than purely technological) aspects. In this context, the *IT Infrastructure Library* (ITIL) has, of all standardization efforts, gained the biggest popularity and can – at least in Europe – now indeed be called a de-facto standard. In its core titles *Service Support* and *Service Delivery*, ITIL provides "best practice" guidelines for IT Service Management.

Implementing *Configuration Management*, a central process of Service Support [1], is often considered the biggest stumbling block in ITIL realization. This is not just because the Configuration Management process itself is not as structured as the other Service Support processes [2]. It is defining the scope and structure of the CMDB, as well as filling and maintaining it, which often proves to be exceedingly difficult and time-consuming. Thus, one of the main challenges in developing solutions for supporting the application of ITIL is to detail the CMDB concept in order to facilitate its implementation.

In the area of systems management, however, integration efforts have given rise to a number of standardized information and data models. Essentially, these models provide means to build a *Management Information Base* (MIB) to be used by IT infrastructure management systems – i.e. a MIB is a logical view on the management-relevant aspects of a part of an IT infrastructure. Thus, at first glance, a CMDB could be seen as not much else than yet another MIB (or a collection of MIBs).

The remainder of this paper is structured as follows: Sec. 2 defines the usage of important terms in the context of this paper and illustrates correlations between the terms used in a CMDB context and those used in a MIB context. This leads to the interesting question: Are existing techniques (management models) capable of solving some of the problems occurring when setting up a CMDB? The answer is based on examining core CMDB concepts (Sec. 3), defining basic criteria for a potential CMDB model, and applying these criteria to some of the most common management models (Sec. 4). Sec. 5 concludes by further investigating the commonalities and differences of the concepts underlying CMDBs and MIBs and discussing possible integration approaches.

2 From Managed Objects to Configuration Items

When discussing management models or ITIL Configuration Management, many terms mean different things to different people. The following will define the usage of some essential terms for the context of this paper.

The *Managed Object* (MO) concept has been defined in the OSI Management Framework [3] and been referred to in various architectures for network and systems management. MOs represent the management view of resources, i.e. they abstract from resources (components) in a managed IT infrastructure. They can be seen as an abstract model of a resource or as the data record used to express this model[1]. The set of MOs associated with a system constitutes that system's *Management Information Base* (MIB). MOs that share the same definition are instances of the same *Managed Object Class* (MOC) [4].

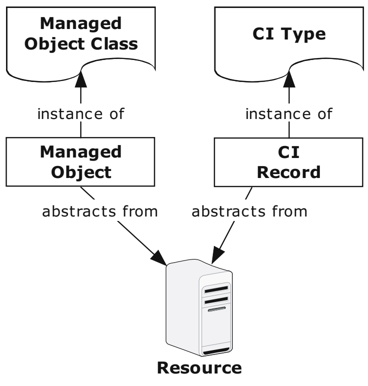


Fig. 1. CIs vs. MOs

Basically, a management *Data Model* (DM) contains MO definitions (or MOCs) in a formalized and detailed enough way to enable a straightforward MIB implementation. Consequently, a DM is bound to a Data Model Language (DML) which defines the syntax in which MOCs are described. If the DML prescribes *how* management information is represented in the DM, the *Information Model*[2] (IM) defines *what* this management information should comprise. In other

[1] In common usage, the term "MO" can also denote the managed component itself.

[2] In the context of this paper the term "Management Model" will be used to refer to the composite model that the union of IM, DML and DM builds (cp. Sec. [5]), but note that in a broader context this is often referred to as the "Information Model" of a management architecture [6].

words, a DM is formalizing and detailing the concepts contained in an IM – consequently various DMs can be derived from a single IM. In an IM context an MO would be an abstract model – in a DM context, an MO would be a data record representing that model. Analogous, in an IM context, MOC is the abstract model of a class of MOs (or a type of MO) – in a DM context, an MOC is a formal definition that is used to instantiate MOs of a specific type (or used to be refined into other MOCs). Most of the standardized management models focus on defining a DM, and document the underlying IM only informally or incompletely [5].

The concept of a *Configuration Item* (CI) in the context of ITIL's guidance on *Configuration Management* (CM) seems similar to that of an MO. A CI is a component of an IT infrastructure (or other items associated with that infrastructure) which is put under the control of the Configuration Management process [1]. Relevant information on CIs (CI attributes) and the relationships between CIs are to be recorded in the *Configuration Management Database* (CMDB). ITIL does not make a clear distinction between the CM view (model) of a component or its expression in the documented CI attributes and the component itself. In this paper, the term *CI Record* (CIR) will be used to refer to the data record contained in the CMDB, while CI will denote the abstract model (CM view) of the component. As depicted in Fig. 1, a CI is abstracting from a resource much in the same manner as an MO. According to ITIL, CIs should be classified into *CI types*. ITIL takes a broader view on what should be regarded as a CI (including e.g. documentation and services defined in SLAs) than what is usually referred to as an MO. Also a CMDB seems to cover a larger part of an IT infrastructure than what one usually thinks of as a "system" (the scope of the management information stored in a MIB). However, since CI records often describe the same kind of IT resources (like software and hardware items) as MOs, there is an obvious analogy between the concepts of MO and CI/CIR, MOC and CI type as well as MIB and CMDB.

Given the existence of standardized management models for network and systems management that define a DML, as well as DMs with large numbers of pre-defined MOCs, it surprises that equivalent standards for building CMDBs have not been proposed so far. This begs the question whether some of the existing management models might be useful for filling this gap. Before this, the concept and purpose of a CMDB needs to be analyzed in more detail.

3 The CMDB Idea

Unfortunately, the guidance ITIL itself gives on the CMDB is neither comprehensive nor consistent in all details. Consequently, it is difficult to give a compendious definition of it. Basically there are two views on the CMDB within ITIL which are not necessarily conflicting, but still address distinct aspects. On the one hand, the CMDB is a logical model of the IT infrastructure and IT services whose creation and maintenance is the main deliverable of the *Configuration Management* process, as discussed in the according chapter in Service Support [1]. On the other hand, the CMDB is seen as a sort of "information hub": In most cases, whenever an ITIL service management process needs to access information outside its immediate scope of responsibility, this is supposed to happen through querying the CMDB. This information can refer to things quite different from IT infrastructure elements or services, e.g. artifacts of other ITIL processes like incident records, but also records on information like customer and user data, whose control is usually not within the scope of IT management. ITIL itself makes no definitive statements on this duality of the CMDB (or even acknowledges it expressively), but it is up to discussion in any real-world implementation whether Configuration Management needs to maintain the latter kind of information – or just ensure access to it. Besides demanding the documentation of relations and dependencies between CIs (including containment relations), ITIL gives almost no guidance on CMDB implementation specifics.

For Configuration Management ITIL defines five[3] basic activities [1,7]: *Configuration Management Planning* (defining scope, purpose, responsibilities etc. of Configuration Management), *Configuration Identification* (defining what CIs are to be included into the CMDB and in what form), *Control of CIs* (assuring up-to-date recording of the characteristics of CIs – in particular in the event of changes), *Configuration Status Accounting* (reporting and control of the current version and change history of all CIs) and*Configuration Verification and Audit* (Planning and carrying out configuration audits to verify accuracy and completeness of the CMDB).

The first two activities listed could be seen as parts of a CMDB setup project. In the terms of Sec. 2 main tasks of such a project would include defining an IM for the CMDB, choosing a suitable DML and a DM. Note however that *Continuous Improvement* is a central concept underlying all ITIL guidance, and consequently, these two activities might be repeated in its context. An important conclusion can be drawn from this: The requirements for the information contained in the CMDB are scenario specific (IT organization specific), can change, and consequently the IM underlying a CMDB might need to be adapted (see also Sec. 4.1). This does not mean however that the requirements on the IM will differ vastly between two providers of similar IT services – there should be large intersections that could be used for building "Base Information Models" for CMDBs.

"Configuration Status Accounting" is a concept adopted from Software Configuration Management that refers to keeping track of the life cycle status of a CI (what version of a specific CI is "in testing", "in operation" etc.). This implies that infrastructure elements should be tracked in a CMDB even before they come into operation. Also, as can be seen from the activities "Control of CIs" and "Configuration Verification and Audit", the concept of ITIL Configuration Management per se assumes more or less manual maintenance of the CMDB and does not rely on any technology to feed data into it. In a ITIL-aligned organization this is not necessarily as complex as one might assume. In day-to-day operations, the only modifications to the content of a CMDB should be triggered by activities of the Change Management process. Results from automatic infrastructure scans can be extremely useful for "Configuration Verification and Audit", but should not be fed unreviewed into the CMDB. Still, this implies that the practice of CMDB maintenance stays comparatively labor-intensive and rises with the amount of information stored in the CMDB. It is therefore important to find the appropriate level of detail in which CIs are to be recorded in the CMDB, achieving a balance

[3] Seven on some counts, but we disregard like most other literature the very generic "CMDB back-ups, archives and housekeeping" and "Providing a Configuration Management service".

between the benefits of information availability and the resources and effort needed to support it [1]. So in summary, a CMDB

– exits to serve the essential information needs of the ITSM processes defined in ITIL
– should be kept "slim" and closely aligned with these information requirements
– contains a model of the IT infrastructure and services
– documents relations between any CIs.

But even though the concept of a CMDB is different from that of a MIB (see Sec. 5), at their core they still both model (parts of) IT infrastructures. ITIL neither gives concrete instruction towards implementing a CMDB, nor have standardized models been established. This begs the question, whether existing information models can be used or adapted for building CMDBs.

4 On Reusing Management Models

Even in the few instances where criteria for CMDB tools are discussed, these efforts are usually focussed on functional requirements (e.g. visualization) and integration with other databases [8,9]. Limiting assessments only to these requirements however bears the danger of not addressing key standardization issues for CMDBs. An effective and sustainable solution to CMDB integration issues will need to be based on at least some partial standardization on the level of a CMDB IM, DM and DML. Consequently, the criteria outlined below are a first step towards documentation of requirements for the design (or selection) of an IM, DM and DML that a CMDB tool will explicitly or implicitly have to be based upon. In Sec. 4.2 the proposed criteria are applied to three existing management models to evaluate the possibilities of reusing their concepts.

4.1 Requirements on a Management Model Reusable for Building a CMDB

With the CMDB concept being rather ambitious, a management model for a CMDB will need to fulfill a number of requirements. Note that many requirements cannot be exactly mapped to either DM, DML, or IM, as there are many interdependencies between these aspects (e.g. a very simple DML might not be able to express complex IM concepts) as well as between them and implementation or architecture specifics (e.g. will the CMDB comprise several physical databases, has Configuration Management a say in how information is stored and accessed in other enterprise databases?).

Adaptability of Model - All ITSM processes are subject to *Service Improvement Programs* (Continuous Improvement, cp. Sec. 3). Consequently, the CMDB, subject to *Continuous Improvement* as well, must be capable of dealing with changing requirements, especially regarding scope, nature and level of detail of the documented information. To keep the costs of adapting the IM low, the DML should allow easy extensibility of the DM.

Alignment to ITSM information needs - Obviously, the IM for CMDB should address all the information requirements of the ITSM processes and consequently include models of all relevant entities (including e.g. Incident Records etc.). On the other hand, to keep the CMDB in principle “human maintainable”, it should not cover too much information or aspects which are not essential in this context.

Comprehensive view on infrastructure and component relations - The documentation of CI relationships (e.g. for service impact analysis) is maybe the single most essential concept in the CMDB context. Consequently, the IM should include basic relations between common CI types and the DML should support modeling multiple, preferably even user-definable, relationships between CIs.

Inclusion of ITSM process artifacts - This criterion refers to the “information hub” nature of the CMDB (cp. Sec. 3). Each ITSM process defined in ITIL not only has specific requirements about what information should be contained in a CMDB, it also creates (ITSM) data itself, i.e. process artifacts such as incident records. Analogous to above criterion, the relationships among these process artifacts (e.g. what incident records are linked to a specific problem record?) as well as between them and infrastructure CIs (e.g. what problem records are associated with a specific infrastructure CI?) need to be included in the CMDB and in consequence should be part of the IM. As there is no convincing argument for putting all process artifacts under Configuration Management control in the same way it is done with infrastructure CIs – and these process records are usually controlled through process-specific tools (e.g. Incident Management System) – it seems likely that in a real-world CMDB implementation these artifacts and infrastructure CIs will be stored in separate physical databases. In that case, a common DML suited for building models of infrastructure components as well as process artifacts could ensure obstacle-free integration.

Integration with external databases - Information of relevance for ITSM might be managed and stored outside the IT organization itself – either in enterprise databases (for example employee data managed by the Human Resources department) or in external CMDBs (e.g. of an external IT sub-service provider). This information will most likely be outside Configuration Management control, and consequently so will be the technical nature of access methods. The DML and DM should therefore lend themselves to integration with a variety of data sources (e.g. by providing easy XML/Web Services mappings).

Integration with (other) network and systems management data stores - Much of the information that should be contained in a CMDB cannot be gathered solely by using resource management or discovery tools (e.g. systems’ locations, compositions of services). However, for information aspects that can be discovered using management tools, verification and audit of CMDB records can benefit greatly from integration with these tools. A DML for Configuration Management should therefore ease reconciliation of data stored in the CMDB with that of other existing management systems (e.g. by providing mappings to other common DMLs).

Support for life cycle status accounting - ITIL demands that the life cycle status of any CI is tracked and documented. This should be reflected in the IM. Also information pertaining to all life cycle phases should be accessible through a CMDB – via attributes or relationships to other CIs and data records (e.g. acquisition date, test records, etc.).

Catalog of basic CI types - Provisioning of common CI types (or MOCs) (informally in the IM – though preferably in the form of an extendable but ready-to-use DM) could significantly shorten the time-to-implementation for a CMDB.

4.2 Assessment of Current Management Models

The apparent similarity of the MO and CI concepts prompts for a reexamination of existing management models. By applying the criteria catalogue presented above, we assess state of the art management models regarding their possible reuse building CMDBs.

Fig. 2 gives a comprehensive overview of the criteria catalog and illustrates how different approaches fulfill these criteria. In the following, we discuss the *Internet Management Model* (IMM) [10], the *Common Information Model* (CIM) [11] and the *Shared Information/Data Model* (CIM) [12].

Internet Management Model (IETF). The Internet Management architecture has two main pillars: The *Simple Network Management Protocol* (SNMP) and a large number of *MIB modules* published in RFCs [5]. The latter build what, though there is no official name for it, could be called the *Internet Management Model* (IMM) that covers a lot of system types in its scope. Following its original design goal of providing a simple way to manage network resources, this model's focus is comparatively narrow. MIB modules contain the MO definitions for a specific type of system – but in IMM an MO often represents a very small aspect of a system that one would generally rather think of as an attribute, e.g. an MO can be a single counter variable. In the Internet Management architecture MOs/MIBs are intended to be stored on the managed system and accessed remotely via an SNMP agent. In the terms defined in Sec. 2, a MIB module (e.g. for a type of switch) could be seen as a data model MOC representing a type of system.

Documenting relationships between MOs is not supported by IMM (except containment within a single system's scope) and as an IMM-MIB is limited to describing a single system, a view on the entire infrastructure and the relationships between its compoments is not supported by the model. In practice this gap is often filled by functionality provided by SNMP-based network management tools (management platforms) that for example support viewing network topologies. Also, an Internet MIB is concerned only with the operational state of a resource. IMM is not designed to support tracking the lifecycle status of a resource.

With the Internet Management architecture being the first widely adopted and implemented management standard, integration with other standards was at the time of its conception of no concern – though concepts for integrating it with later management architectures have been designed. Also, due to its technical focus, support for integration with enterprise databases or linking to documents (process artifacts) was never intended in IMM. The concept of IMM presumes the requirements of management to be rather static. It is not intended that MIB modules can be customized by the user (i.e. the operator of the infrastructure), e.g. for addressing operator-specific or changing requirements.In practice, it is again management platforms that address this gap by filtering or consolidating information or allowing to retrofit the infrastructure view gathered from the MIBs with manually added information.

Common Information Model. The Common Information Model (CIM) [11] is an object-oriented management model that aims at providing a common way to represent

	IMM	CIM	SID
Adaptability of Model	✗	✓	✓
Alignment to ITSM information needs	✗	☑	☑
Comprehensive view	✗	✓	✓
ITSM process artifacts	✗	✗	✓
Integration with external databases	✗	✗	–
Integration with management data stores	☑	☑	–
Support for life cycle status accounting	✗	☑	☑
Catalog of basic CI types	✓	✓	☑

Legend	
✓	satisfied
☑	partially satisfied
✗	not satisfied
–	not applicable

IMM: Internet Management Model CIM: Common Information Model SID: Shared Information/Data Model

Fig. 2. Assessment of management models

information about networks and systems as well as services. It defines managed resources as object classes that can be further refined by means of strict inheritance. Part of CIM is textual, human-readable language (*Managed Object Format (MOF)*[13]) for describing modeling constructs that can be processed by automated tools. Since all CIM classes derive from the `managed element` class as defined in the Core Model, CIM provides a coherent view on the modeled infrastructure. This view, however, does not include the linkage of ITSM processes to infrastructure elements. In particular, key concepts of ITIL such as Incident records fall out of CIM's scope.

CIM makes extensive use of relationships, namely associations and aggregations; together with its object-oriented approach this yields a sufficient level of expressiveness and extensibility. CIM features a large amount of standardized object definitions. In total, the amount of managed object classes defined in CIM comes close to 900. It is therefore fair to say that CIM represents a solid basis for integrated management, but is a fairly complex model. Understanding the relationships between these classes and adapting CIM to the needs of an organization requires a serious effort [14]. Moreover, much of the information conveyed in these classes deals with low level details of resources and clearly exceeds the ITSM scope.

Despite some exceptions within the CIM application schema, CIM's focus is on the operation phase of an IT infrastructure. Thus, it provides only rudimentary support for life cycle phases such as planning.

While CIM provides a rich data model, it currently shows deficits in expressing business-oriented concepts as required by ITSM. In particular, only few MOCs for expressing service-related management information have been defined so far. This might be due to the fact that it has its roots in the area of desktop systems and has over time developed into a more generic model.

Shared Information/Data Model. Compared to CIM and IMM, SID (Shared Information/Data Model) [12,15] is less centered around DM concepts, but provides an information model. SID is an integral part of the NGOSS (New Generation Operations Systems and Software) initiative by TMF (TeleManagement Forum). Including SID in this assessment thus bears some ambiguities. However, SID is the first information model tightly coupled to management processes. If there was a CMDB for the eTOM process framework, it would be based on SID.

The SID model employs an object-oriented modeling approach and draws a clear distinction between the system and business view on management information. Accordingly, it is organized into System and Business domains, which are in turn partitioned into Aggregate System Entities (ASEs) respectively Aggregate Business Entities (ABEs). ASEs are intended to facilitate linkage between business and system view, since they elaborate on the concepts defined in ABEs. SID is strongly tied to the eTOM Process Framework [16] in that Business Domains accord with eTOM Level 0 concepts. However, since eTOM and ITIL vary considerably in structure, SID's ABEs will not be suitable for unqualified inclusion into a CMDB information model.

The concept of *Continuous Improvement* is not explicitly addressed in eTOM and NGOSS. It is therefore not quite clear yet wether user/operator adaptability will be a central design goal for future implementations of a SID data model. Inherently, the SID model is specified as a rooted class hierarchy. It makes use of object-oriented concepts like generalization, associations, aggregations and compositions to express relationships between entities and is thus able to provide a coherent view on the IT infrastructure.

With entities and attributes being described by a mixture of descriptive text, UML diagrams and tables, SID also provides a reasonable level of expressiveness. This includes the use of finite-state machines to model life-cycle aspects – a concept that has been incorporated from DEN-ng.

SID's strength clearly lies in its modeling of higer-level concepts (e.g. *Service*, *SLA*), where most MOCs have been defined. While in this regard SID offers considerable benefits over IMM and CIM in terms of maturity, it currently defines only few MOCs for expressing low level details of resources. While SID's focus is clearly on eTOM processes, it exhibits a number of sound concepts that a CMDB information model would benefit from. This includes the coupling of model entities and business processes to provide a business viewpoint on the data/information.

The detailed investigation of three different concepts for information modeling showed that none of them is directly applicable to build a CMDB. For instance IMM and CIM do not offer any support to model the life cycle dependence of CIs. All approaches lack a strict focus on ITSM. Besides SID, no information model offers standard, easy to deploy MOCs for expressing higher-level concepts such as services. However, it shows deficits in modeling low-level parameters of system and network components – whereas CIM and IMM feature a large amount of standard MOCs for that purpose. Fig. 2 comprehensively illustrates these gaps towards implementing a CMDB based on well established information models.

5 The CMDB-MIB Gap and First Steps on Bridging It

As seen in Sec. 4.2, the established management models do not lend themselves to immediate application for CMDB design. The reason for this apparently lies in some fundamental differences between the design goals and design philosophies for MIBs and CMDBs. The purposes a CMDB and a MIB serve are, despite seeming similar on the surface ("providing a model for IT management"), quite distinct. Some distinctions were already touched upon in previous sections. This section outlines some additional aspects before continuing to discuss approaches to integrate MIBs and CMDBs.

5.1 Understanding Differences and Similarities – The CMDB-MIB Gap

A MIB serves to make the tasks of day-to-day operations easier for operators and administrators by addressing the challenges of infrastructure diversity and (physical/geographical) distribution. In contrast, a CMDB serves decision makers in the various ITIL service management processes for which a comprehensive overview (e.g. on the factors influencing service performance) is often more valuable than minute detail.

Also the philosophy of who has the last say about the contents of a CMDB or MIB differs. A MIB usually comes with the system it models, i.e. its design is done by the system vendor. A later customization by (management) users on a model level is not intended. Realizing adapted models is often possible through functionality of management systems – these are then usually stored in a format proprietary to that management system. ITIL's "adopt and adapt" philosophy by contrast suggests that the scope and structure of a CMDB should be adaptable to cope with changing scenario-specific requirements.

Also, while tracking components through all life cycle phases, CMDBs do not provide up-to-the-minute data of the operational status of components – nor is a CMDB intended to be a tool to manipulate that status with effect onto the live environment. Even though editing and auditing the CMDB can be made more efficient by tools, the maintenance of the CMDB as a whole is an organizational practice independent from any technology, and can consequently probably never be 100% automatized.

So, a CMDB is really a different beast than a MIB. Both emphasize in their models distinctly different aspects of the IT infrastructure as they serve to support different tasks by different groups of stakeholders. But then again, they both offer an abstracted management view of the same infrastructure.

There is an analogy to that in IT system modeling. It is commonly accepted that there is a need for different model types in software and systems engineering. As it is good practice in this discipline, there should be mechanisms for ensuring consistency between MIB and CMDB models and facilitate reuse of shared information.

5.2 Further Directions – Bridging the CMDB-MIB Gap

Some ITIL consultants might argue that the requirements for the model underlying a CMDB are dependent on the individual characteristics of every IT organization, and therefore no common model will fit all scenarios. However, the growing adoption of ITIL (and related standards like ISO 20000) furthers a basic organizational standardization – while on the infrastructure side, standardization of hardware and software has also been evolving. It therefore seems very unlikely that the requirements for a CMDB will vary so enormously across IT organizations that are comparable in size and offered services. Given the very high level of abstraction and generality in ITIL's CMDB guidance, there is ample room (and need) for a common CMDB "reference model", comprising an IM, DML and DM that are applicable to a large number of CMDB scenarios (and adaptable to suit most).

While a sound approach for such a standardized model for ITIL is currently missing, the TMF has pursued a similar goal with SID (cp. Sec. 4.2). Although SID is built around eTOM processes and therefore not directly applicable to ITIL-based ITSM, many of its concepts seem to be general enough in design to be suitable for reuse in CMDB models.

Towards this goal, we are working on the definition of a CMDB reference model. The first step is to define a set of CI types which are indispensable for ITIL-based IT Service Management and the relationships between them. (e.g. *ServiceIncident, ServiceProblem*). When defining CI Types for common kinds of infrastructure components or business entities, we try to reuse existing (abstract) information model concepts from CIM and SID. Information items that have not been covered by these approaches, such as ITIL process artifacts, are conceptualised using reference processes we concurrently develop based on ITIL process descriptions (cp. [2]). In particular, this includes the information flow between processes – an aspect only incompletely addressed by ITIL for some processes – and basic *Key Performance Indicators* (KPIs).

Given the differences between MOs and CIs, it seems likely that MIBs and CMDBs will coexist in most IT organizations in the foreseeable future. It would be advantageous to have means of exchanging information between them, e.g. for facilitating CMDB verification and audit. After all they both contain models of the same infrastructure. For this, the ability to define mappings between the MIB and CMDB data models (necessarily based on IM and DML mappings) would be desirable. But we hope that also information flow in the opposite direction could be an asset. If service dependencies maintained in a CMDB can be integrated with the real-time data out of MIBs, this might be a step forward towards the ambitious goal of tracking the operational status of a service in a “Service MIB” (cp. [17]).

6 Conclusion

This paper presented a comparison of the common notion of a MIB against the yet evolving idea of a CMDB. An assessment of established management models showed that their concepts cannot be reused unadapted for building a CMDB. The cause for this lies in a fundamental difference between the principles underlying both concepts, stemming primarily from the distinct goals pursued in their design.

In the light of ITIL-based ISO 20000 certifications for IT service providers gaining momentum, an approach for building capable and sustainable CDMBs is dearly needed. This is achieved best not by retrofitting existing tools, but by approaching CMDB design “top-down” – based on a sound requirements analysis and the development and standardization of appropriate models. In addition, given the existence of MIB-based management systems in most larger IT organizations, there is a strong point for arguing that a CMDB should be closely integrated with these. Approaching such an integration should start at the model level as well and not be based on “bottom-up” application integration.

Acknowledgment

The authors wish to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of this paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich (LMU), the Technical University of Munich (TUM), the University of the Federal Armed Forces Munich, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. Their web-server is located at http://www.mnm-team.org. This paper was supported in part by the EC IST-EMANICS Network of Excellence (#26854).

References

1. Office of Government Commerce (OGC), ed.: Service Support. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK (2000)
2. Brenner, M.: Classifying ITIL Processes — A Taxonomy under Tool Support Aspects. In: First IEEE/IFIP Workshop on Business–Driven IT Management (BDIM 06), IEEE (2006)
3. ISO/IEC: Management Framwork. (1989) ISO 7498-4.
4. ISO/IEC: Management Information Model. (1993) ISO 10165-1.
5. Pras, A., Schoenwaelder, J.: On the difference between information models and data models. Technical report, IETF (2003) RFC 3444.
6. Hegering, H.G., Abeck, S., Neumair, B.: Integrated Management of Networked Systems. Morgan Kaufmann Publishers (1999)
7. OGC, ed.: Introduction to ITIL. IT Infrastructure Library. The Stationary Office (2005)
8. Colville, R.J.: CMDB or Configuration Database – Know the Difference. Gartner. (2006) Research Note G00137125.
9. Pink Elephant: PinkVerify – Configuration Management Certification Criteria. (2006) `http://www.pinkelephant.com/en-GB/PinkVerify/`.
10. Case, J., McCloghrie, K., Rose, M., Waldbusser, S.: RFC 1902: Structure of Management Information for SNMP V2. (1996)
11. DMTF: Common Information Model (CIM) Version 2.9. (2005)
12. TMF: Shared Information/Data (SID) Model Concepts, Principles, and Domains. TMF. (2006) GB922.
13. DMTF: Core MOF Specification 2.1. (1998)
14. Alexander Keller, Heather Kreger, K.S.: Towards a CIM schema for runtime application management. In: International Workshop on Distributed Systems: Operations & Management (DSOM 2001), IFIP/IEEE (2001)
15. Strassner, J.: Policy based network management. Morgan Kaufmann Publishers (2004)
16. TMF: enhanced Telecom Operations Map (eTOM), The Business Process Framework For The Information and Communications Services Industry. (2002) GB921.
17. Sailer, M.: Towards a Service Management Information Base. In: IBM PhD Student Symposium at ICSOC05, Amsterdam, Netherlands (2005)

Author Index