Jürgen Schönwälder
Joan Serrat (Eds.)

LNCS 3775

Ambient Networks

16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2005
Barcelona, Spain, October 2005, Proceedings

ifip

Springer

Jürgen Schönwälder   Joan Serrat (Eds.)

Ambient Networks

16th IFIP/IEEE International Workshop
on Distributed Systems: Operations and Management, DSOM 2005
Barcelona, Spain, October 24-26, 2005
Proceedings

Springer

Volume Editors

Jürgen Schönwälder
International University Bremen
P.O. Box 750 561, 28725 Bremen,Germany
E-mail: j.schoenwaelder@iu-bremen.de

Joan Serrat
Universitat Autonòma de Barcelona
Computer Vision Center
Edifici O, 08193 Cerdanyola, Spain
E-mail: joan.serrat@uab.es

Library of Congress Control Number: 2005934373

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

ISSN 0302-9743
ISBN-10 3-540-29388-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-29388-0 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11568285 06/3142 5 4 3 2 1 0

Preface

This volume of the Lecture Notes in Computer Science series contains all the papers accepted for presentation at the 16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2005), which was held at the University Politècnica de Catalunya, Barcelona during October 24–26, 2005.

DSOM 2005 was the sixteenth workshop in a series of annual workshop and it followed the footsteps of highly successful previous meetings, the most recent of which were held in Davis, USA (DSOM 2004), Heidelberg, Germany (DSOM 2003), Montreal, Canada (DSOM 2002), Nancy, France (DSOM 2001), and Austin, USA (DSOM 2000). The goal of the DSOM workshop 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 this field. In contrast to the larger management symposia, such as IM (Integrated Management) and NOMS (Network Operations and Management Symposium), the DSOM workshops are organized as single-track programs in order to stimulate interaction among participants.

The focus of DSOM 2005 was "Management of Ambient Networks". Ambient networks is a new vision to provide accessibility and distributed services through the dynamic composition of networks. The wide adoption of packet switched networking technologies and the fast growing wireless networking infrastructures in public as well as in private spaces allow systems to choose how to obtain connectivity. Systems may also dynamically form new networks and the devices or the whole network may be mobile. Furthermore, many ambient networks will be in private spaces, owned and "operated" by non-technical users (home networks). The heterogeneity of the services and resources participating in ambient networks and the dynamics associated with the composition of networks poses new management challenges. While some papers presented at the workshop address some of these challenges, there was also room for papers addressing general topics related to the management of distributed systems.

This year, DSOM 2005 was for the first time co-located with several related events, namely the 8th International Conference on Management of Multimedia Networks and Services (MMNS 2005), the 5th IEEE International Workshop on IP Operations & Management (IPOM 2005), the 2005 Symposium on Self-Stabilizing Systems (SSS 2005), and the 1st IEEE/IFIP International Workshop on Autonomic Grid Networking and Management (AGNM 2005). All these events together formed the 1st International Week on Management of Networks and Services (MANWEEK 2005).

DSOM 2005 attracted a total of 87 papers with authors from 23 different countries. Every submitted paper received at least three reviews. The authors were invited to write a rebuttal to the reviews while the members of the Technical

Program Committee discussed the papers online. The final paper selection was based on the reviews, the author's feedback and the online discussions within the Technical Program Committee. Out of the 87 submitted papers, 23 were finally accepted for presentation in eight paper sessions.

This workshop owes its success to all members of the Technical Program Committee who did a great job of encouraging their colleagues to submit high-quality papers. The Technical Program Committee members and all the reviewers also deserve special thanks for their constructive and detailed reviews, which were key to assuring the high quality of the workshop. We also thank Lisandro Zambenedetti Granville for running the online paper submission system JEMS which again proved to be invaluable.

Last but not least, we thank all sponsors and patrons who helped to make DSOM 2005 a success and a very enjoyable experience.

August 2005

Jürgen Schönwälder
Joan Serrat

Organization

Conference Chairs

Jürgen Schönwälder	*International University Bremen, Germany*
Joan Serrat	*University Politècnica de Catalunya, Spain*

Local Arrangements

Joan Serrat	*University Politècnica de Catalunya, Spain*

Technical Program Committee

Ehab Al-Shaer	*DePaul University, USA*
Nevil Brownlee	*University of Auckland, New Zealand*
Raouf Boutaba	*University of Waterloo, Canada*
Marcus Brunner	*NEC Europe Ltd., Germany*
Mark Burgess	*Oslo University College, Norway*
Omar Cherkaoui	*University of Quebec in Montreal, Canada*
Alexander Clemm	*Cisco Systems, USA*
Luca Deri	*ntop.org, Italy*
Gabi Dreo Rodosek	*University of Federal Armed Forces Munich, Germany*
Metin Feridun	*IBM Research, Switzerland*
Olivier Festor	*LORIA – INRIA Lorraine, France*
Alex Galis	*University College London, UK*
Lisandro Z. Granville	*Federal University of Rio Grande do Sul, Brazil*
Takeo Hamada	*Fujitsu Labs of America, USA*
Heinz-Gerd Hegering	*Leibniz Supercomputing Center, Germany*
Joseph Hellerstein	*IBM T.J. Watson Research Center, USA*
James Hong	*POSTECH, Korea*
Cynthia Hood	*Illinois Institute of Technology, USA*
Gabriel Jakobson	*Altusys Corp., USA*
Alexander Keller	*IBM T.J. Watson Research Center, USA*
Yoshiaki Kiriha	*NEC, Japan*
Lundy Lewis	*Southern New Hampshire University, 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 in Montreal, Canada*
Jose Luis Marzo	*University de Girona, Spain*

Jose Marcos Nogueira	*Federal University of Minas Gerais, Brazil*
George Pavlou	*University of Surrey, UK*
Aiko Pras	*University of Twente, The Netherlands*
Jürgen Quittek	*NEC Europe Ltd., Germany*
J. Christopher Ramming	*SRI International; DARPA, USA*
Danny Raz	*Technion, Israel*
Akhil Sahai	*HP Laboratories, USA*
Adarsh Sethi	*University of Delaware, USA*
Rolf Stadler	*KTH Royal Institute of Technology, Sweden*
Burkhard Stiller	*University of Zurich and ETH Zurich, Switzerland*
Joe Sventek	*University of Glasgow, UK*
Radu State	*LORIA – INRIA Lorraine, France*
Frank Strauß	*TU Braunschweig, Germany*
John Vicente	*Intel, USA*
Victor Villagrá	*University Politécnica de Madrid, Spain*
Vincent Wade	*Trinity College Dublin, Ireland*
Felix Wu	*University of California at Davis, USA*
Makoto Yoshida	*University of Tokyo, Japan*

Additional Reviewers

Constantin Adam	*KTH Royal Institute of Technology, Sweden*
Anatoly Andrianove	*DePaul University, USA*
Bassam Aoun	*University of Waterloo, Canada*
Marco Ballette	*University of Essex, UK*
Eusebi Calle	*Universitat de Girona, Spain*
Adel El Atawy	*DePaul University, USA*
Lluis Fabrega	*Universitat de Girona, Spain*
David Fernandez	*Universidad Politécnica de Madrid, Spain*
Carlos Mauricio Figueiredo	*Federal University of Minas Gerais, Brazil*
Joao Girao	*NEC Europe Ltd., Germany*
Alberto Gonzalez	*KTH Royal Institute of Technology, Sweden*
Hazem Hamed	*DePaul University, USA*
David Hausheer	*ETH Zurich, Switzerland*
Vasil Hnatyshin	*Rowan University, USA*
Roy Ho	*University of Surrey, UK*
Khaled Ibrahim	*DePaul University, USA*
Brent Ishibashi	*University of Waterloo, Canada*
Teodor Jove	*Universitat de Girona, Spain*
Hiroaki Kamoda	*NTT DATA Corporation*
Torsten Klie	*L3S Research Center, Germany*
Noura Limam	*University of Waterloo, Canada*
Apostolos Malatras	*University of Surrey, UK*
Cesar A Mantilla	*Universitat de Girona, Spain*

Shin'ichiro Matsuo	*NTT DATA Corporation*
Loubna Mekouar	*University of Waterloo, Canada*
Giorgio Nunzi	*NEC Europe Ltd., Germany*
Carmelo Ragusa	*Southampton University, UK*
Helmut Reiser	*University of Munich, Germany*
Lopa Roychudhari	*DePaul University, USA*
Nina Saxena	*Intel Corporation, USA*
Stefan Schmid	*NEC Europe Ltd., Germany*
Fabricio Silva	*Federal University of Minas Gerais, Brazil*
Sharad Singhal	*HP Laboratories, USA*
Siva Sivavakeesar	*University of Surrey, UK*
Yongning Tang	*DePaul University, USA*
Hector Trevino	*Cisco Systems, USA*
Hector Velayos	*KTH Royal Institute of Technology, Sweden*
Pere Vila	*Universitat de Girona, Spain*
Sonia Waharte	*University of Waterloo, Canada*
Fetahi Wuhib	*KTH Royal Institute of Technology, Sweden*
Alvin Yew	*University of Essex, UK*
Bin Zhang	*DePaul University, USA*
Joanna Ziembicki	*University of Waterloo, Canada*

Sponsoring Institutions

Institute of Electrical and Electronics Engineers (IEEE)
IEEE Communications Society (ComSoc)
International Federation for Information Processing (IFIP)
Universitat Politécnica de Catalunya, Spain

Table of Contents

Information Models and Metrics

On the Formalization of the Common Information Model Metaschema
Jorge E. López de Vergara, Víctor A. Villagrá, Julio Berrocal 1

Ontology-Based Integration of Management Behaviour and Information Definitions Using SWRL and OWL
Antonio Guerrero, Víctor A. Villagrá, Jorge E. López de Vergara, Julio Berrocal 12

On the Impact of Management on the Performance of a Managed System: A JMX-Based Management Case Study
Abdelkader Lahmadi, Laurent Andrey, Olivier Festor 24

Security and Privacy

Improving the Configuration Management of Large Network Security Systems
João Porto de Albuquerque, Holger Isenberg, Heiko Krumm, Pauslo Lício de Geus 36

An Architecture for Privacy-Aware Inter-domain Identity Management
Wolfgang Hommel 48

Data on Retention
Ward van Wanrooij, Aiko Pras 60

Policy-Based Management

SLA Design from a Business Perspective
Jacques Sauvé, Filipe Marques, Antão Moura, Marcus Sampaio, João Jornada, Eduardo Radziuk 72

Generic Policy Conflict Handling Using a priori Models
Bernhard Kempter, Vitalian A. Danciu 84

An Approach to Understanding Policy Based on Autonomy and Voluntary Cooperation
Mark Burgess 97

Deployment, Auditing and Tuning

Towards Automated Deployment of Built-to-Order Systems
Akhil Sahai, Calton Pu, Gueyoung Jung, Qinyi Wu, Wenchang Yan, Galen S. Swint 109

A Generic Model and Architecture for Automated Auditing
Hasan, Burkhard Stiller 121

Utilization and SLO-Based Control for Dynamic Sizing of Resource Partitions
Zhikui Wang, Xiaoyun Zhu, Sharad Singhal 133

Performance and Quality of Service

A Decentralized Traffic Management Approach for Ambient Networks Environments
María Ángeles Callejo-Rodríguez, Jorge Andrés-Colás, Gerardo García-de-Blas, Francisco Javier Ramón-Salguero, José Enríquez-Gabeiras 145

Performability Analysis of an Adaptive-Rate Video-Streaming Service in End-to-End QoS Scenarios
I. Martín, J. Alins, Mónica Aguilar-Igartua, Jorge Mata-Díaz 157

Design and Implementation of Performance Policies for SMS Systems
Alberto Gonzalez Prieto, Rolf Stadler 169

Routing

Detection and Diagnosis of Inter-AS Routing Anomalies by Cooperative Intelligent Agents
Osamu Akashi, Atsushi Terauchi, Kensuke Fukuda, Toshio Hirotsu, Mitsuru Maruyama, Toshiharu Sugawara 181

Discovery of BGP MPLS VPNs
Sarit Mukherjee, Tejas Naik, Sampath Rangarajan 193

Policy-Based Adaptive Routing in Autonomous WSNs
Carlos M.S. Figueiredo, Aldri L. dos Santos, Antonio A.F. Loureiro, José M. Nogueira 206

Fault Management

Decentralized Computation of Threshold Crossing Alerts
Fetahi Wuhib, Mads Dam, Rolf Stadler, Alexander Clemm 220

Control Considerations for Scalable Event Processing
Wei Xu, Joseph L. Hellerstein, Bill Kramer, David Patterson 233

Can Dynamic Provisioning and Rejuvenation Systems Coexist in Peace?
Raquel Lopes, Walfredo Cirne, Francisco Brasileiro,
Eduardo Colaço 245

Distributed Management

A Hierarchical Architecture for a Distributed Management of P2P Networks and Services
Guillaume Doyen, Emmanuel Nataf, Olivier Festor 257

Enhancements to Policy Distribution for Control Flow and Looping
Nigel Sheridan-Smith, Tim O'Neill, John Leaney, Mark Hunter 269

Author Index 281

On the Formalization of the Common Information Model Metaschema

Jorge E. López de Vergara[1], Víctor A. Villagrá[2], and Julio Berrocal[2]

[1] Departamento de Ingeniería Informática, Universidad Autónoma de Madrid,
Escuela Politécnica Superior, Francisco Tomás y Valiente, 11, E 28049 Madrid, Spain
jorge.lopez_vergara@uam.es
[2] Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid,
ETSI de Telecomunicación, Ciudad Universitaria, s/n, E 28040 Madrid, Spain
{villagra, berrocal}@dit.upm.es

Abstract. Integrated network management frameworks include a common definition of the managed resources, known as an information model, which is a key factor to describe the domain to be managed. In this scope, it is important to understand the semantics each information model provides to allow interoperation among different integrated management architectures. For this, ontology languages have recently been proposed, because thanks to their formalization they can deal with the semantics of information. Nevertheless, they need to be adapted to meet the management requirements. An alternative to the use of ontology languages can be the formalization of the management information languages to cope with the semantics of the information models. This paper provides a way to formalize one of these management languages: the Common Information Model metaschema. The formalization is based on the use of the Object Constraint Language to define in a formal way the set of natural language rules that describe this metaschema, improving its semantics, comparing also this solution to those based on ontologies.

1 Introduction

Network and service management has been a field in which traditionally proprietary solutions from different vendors were usually imposed. In these solutions the management of those equipments could only be performed with those vendor products. Then, integrated network management architectures appeared that defined standard protocols and information models allowing the interoperability between multiple vendors managers and managed elements.

Due to historical reasons, two different management frameworks have survived the standardization process: Internet network management framework (also known as SNMP, Simple Network Management Protocol) and OSI network management framework (also known as its protocol: CMIP, Common Management Information Protocol). These frameworks are incompatible, so finally each one has got its own application field, even though both frameworks have to coexist in some environments, such as telecommunication companies.

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 1 – 11, 2005.

Later on, other integrated network management architectures have appeared that use other technologies for resources management, different to SNMP or CMIP. The most significant example is the Web-Based Enterprise Management (WBEM) and its associated Common Information Model, CIM.

Each integrated management architecture deals with its own information, defined in a different language: same concepts can be defined to model a resource using incompatible formats, which cannot be directly translated. This issue is a combination of syntax and semantic problems. One way to deal with the semantics of the management information is the use of ontologies: they are formal [1], and thus, the meaning of this information is machine-interpretable.

By applying this knowledge representation technique, the work presented in [2] provided a way to analyze management information languages, being useful to identify their semantic expressiveness. One of the results obtained was that the Common Information Model (CIM) had most of the elements usually contained in ontology languages. However, it was not a model appropriate to deal with the semantics of the information because of its lack of formalization: rules about its structure have been defined in natural language, which is not machine interpretable, so that they cannot be processed and checked.

Then, ontology languages have been proposed to describe the management information [3, 4, 5, 6]. In this case, these languages have a formalized semantics. Still, they have to be adapted to the management scope, as there are some constructs they do not include.

Another way to deal with the semantics of management information is the formalization of the CIM metaschema: in this case a management specific information model is used, and a computer would be able to interpret the information defined in such way. With respect to the formalization of a management language, some works have been found [7, 8], but they are related to GDMO (Guidelines for the Definition of Managed Objects), the language used for OSI Systems Management, which had less constructions in common with ontology languages than CIM, as stated in [2]. The formalization of the CIM metaschema also reinforces the information defined in the CIM schema, which currently includes in its last release more than a thousand classes that base their relationships on that metaschema.

This paper presents an approach to formalize the CIM metaschema. For this, first of all, an analysis of this metaschema is given. Then, it is also compared to UML (Unified Modeling Language) metamodel. Next, a set of rules defined in OCL (Object Constraint Language) are shown that match natural language rules about CIM elements, providing a formalization of the metaschema. After that, this approach is compared with the use of a formal ontology language. Finally, conclusions and future works are also presented.

2 CIM Metaschema Analysis

CIM [9] is the information model defined by DMTF to be used in the Web Based Enterprise Management architecture, and has a considerable acceptation in the industry. This model is object-oriented and much more powerful than SNMP SMI (Structure of Management Information). However, its complexity is lower than

GDMO, as discussed in [2]. With this format, classes can have properties (the name they use for attributes) and methods. Other facets can be defined, thanks to the possibility of specifying new qualifiers [9]. This information model can also be expressed in XML (Extended Markup Language) to exchange the information.

As stated before, CIM has the information model metaschema with a largest number of elements usually included in ontology languages. It includes these characteristics, when comparing it to ontology languages [2]:

- Concepts or classes: They are a collection of instances with the same properties and methods. CIM can define:
  - Metaclasses: This item deals with the possibility of defining classes as instances of other ones. In CIM it is possible to define new statements with qualifiers, which indirectly makes feasible the redefinition of classes.
  - Attributes: Concepts usually have attributes. In CIM they are defined in the local scope of a class and can be instance attributes, class attributes, and polymorph attributes.
  - Facets: Attributes usually have a set of predefined properties or facets. In CIM default value, data type constraint, cardinality constraint, and documentation can be found among other facets such as the access, the key or index, and the identifier. In addition, CIM can define new facets by using qualifiers.
- Taxonomy: Concepts are usually organized in taxonomies, with generalization/ specialization relationships among them. CIM allows the definition of subclasses with simple inheritance.
- Relations and functions: Relations represent a type of interaction between concepts. Functions provide a unique value from a list of valued arguments. CIM can define both relations among classes and functions for every class, with data type constraints.
- Instances: They represent elements of a given concept, a relation or an assertion. CIM allows the definition of class and relation instances.
- Axioms: They model expressions that are always true, and are usually used to define constraints. CIM does not currently support constraints, although a qualifier could be defined with this purpose.

Also, CIM schemas are structured in a similar way to ontology libraries [10]. In this way CIM schemas could be considered an ontology except for their lack of formalism. On the other hand, CIM uses the Unified Modeling Language (UML) class diagrams to model the management information, and several works [11, 12] have identified UML as a valid ontology modeling language.

This set of reasons presents CIM as a good candidate to define management information from a semantic viewpoint. Nevertheless, there is a problem that has to be solved to achieve this goal: as stated before, CIM is not formal (the rules about its metaschema are written in natural language, which cannot be processed and checked by computers), so it is not valid for the definition of heavyweight ontologies. To solve this problem it will be necessary the formalization of its metaschema. For this, the Object Constraint Language (OCL) [13], used in UML to define constraints can be applied, rewriting CIM metaschema rules, avoiding existing ambiguities that are caused because they are currently written in natural language. Other rule languages such as SWRL (Semantic Web Rule Language) [14] would also be useful for this task, but OCL has been chosen because of its integration with UML, and because it is being studied by the DMTF to specify constraints for management classes and objects in the CIM schemas.

3 CIM and UML

CIM semantics has been defined in its metamodel or CIM metaschema, depicted in **Fig. 1**, which describes the elements existing in this model by representing them in a UML class diagram and defining a set of rules about these elements in natural language.

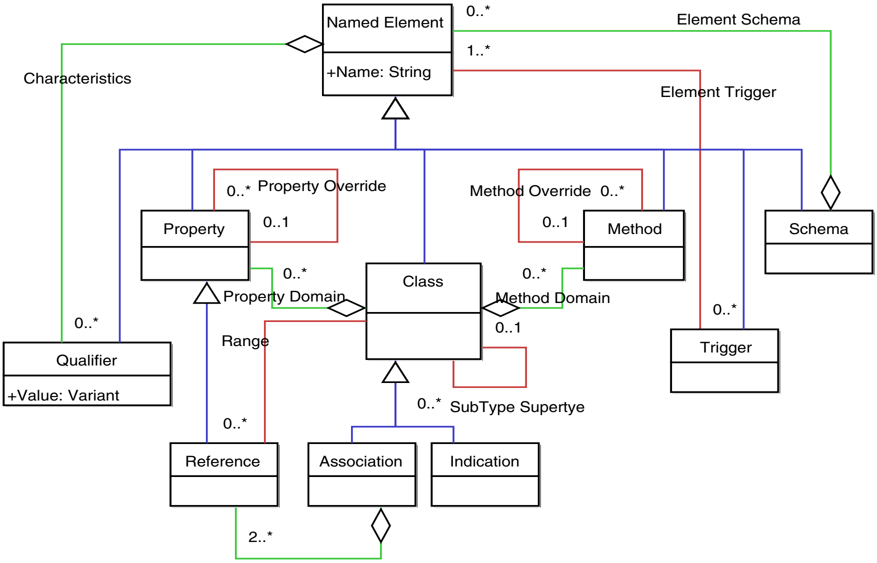


Fig. 1. CIM Metaschema [9]

Given that UML metamodel [15] includes a set of constraints defined in OCL that formalizes the behavior of its elements, a question arises: if CIM metaschema uses UML, are its elements as formal as UML? To answer it, a comparison between CIM and UML is provided, which shows that there are important differences between them.

The first difference is related to abstraction levels: **Table 1** shows a comparison between CIM and the four-layer metamodel architecture used in UML: CIM meta-metamodel (the model used to define the CIM metaschema) is directly UML; the metamodel (the model used to define the models) is the CIM metaschema; the set of CIM schemas are in the model level; finally, CIM schema class instances are the user objects.

If the comparison is focused in the metamodel layer, CIM metaschema is also different to UML metamodel. **Fig. 2** shows a subset of the UML metamodel that includes a set of elements which could be equivalent to CIM metaschema, shown in **Fig. 1**. Although they share a similar structure there is a different number of elements in both figures, as there are more specialization degrees in UML.

Table 1. Comparison of UML and CIM layers

Layer	UML	CIM
Meta-metamodel	OMG MOF meta-metamodel	UML
Metamodel	UML metamodel	CIM Metaschema
Model	UML models	CIM schemas
User objects	UML model instances	CIM class instances

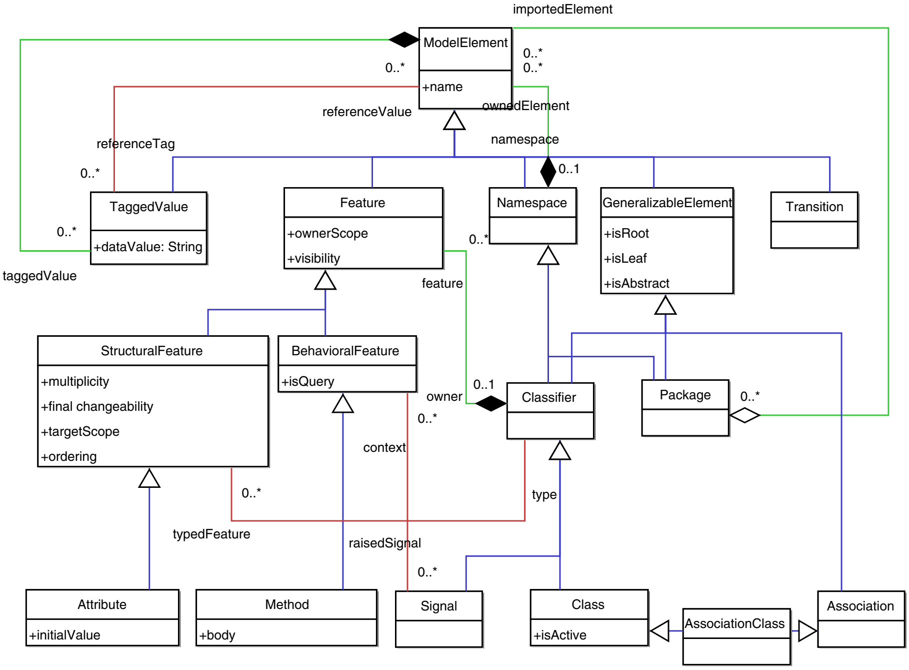


Fig. 2. UML metamodel subset

As a result, UML formalization rules are not directly applicable to CIM metaschema. New rules have to be defined to achieve this goal, as shown in next section.

4 CIM Metaschema Formalization

This section presents a formal specification of CIM metamodel. For this, a set of rules have been defined in OCL, trying to cover the set of rules defined in natural language in CIM specification [9]. During this process, some incongruities were found among existing rules, which were solved when possible. This formalization does not modify existing information defined in CIM, but on the other side it allows its validation with OCL constraints.

To carry on this formalization, all information given in CIM specification has been taken into account: This specification first describes the CIM metaschema with a UML class diagram (as shown in **Fig. 1**). Then, it provides a set of rules written in natural language (and thus, not formal). Next, it specifies in ABNF (Augmented Backus-Naur Form) the MOF (Managed Object Format) syntax. Finally, it presents the CIM metaschema written in MOF format. This specification has been revised by DMTF [16], but defined rules are mostly similar to prior version, and are still in natural language. Some conflicts have been found among these sections:

The UML class diagram that models the CIM metaschema is not complete. It does not include constraints related to each element, neither other elements named in the natural language rules or in the MOF syntax (e.g. Instance element).

Some natural language rules are incongruous, as there exist different properties for an element in different rules (e.g. rules about the Qualifier element).

Other rules are redundant with respect to the class diagram (e.g. cardinality relationships, or element specialization), so that it is not necessary their definition.

Taking into account these conflicts, a formalization has been performed on the CIM metaschema, as shown in **Fig. 3**.

The formalized diagram includes these points:

1. All elements named in rules or in MOF syntax have been added, including those that were not depicted previously (e.g. DataType and Instance elements, and some associations between elements).
2. All association ends have been named when they start and finish in the same element, to improve the diagram semantics and to make easier the definition of OCL rules (e.g. Overriding and Overridden in Property and Method elements, or Subtype and Supertype in Class element). For the rest of associations the name of the association end is directly the name of the associated element, except when the constraint rule uses other name (e.g. Range in the association end of Class with Reference, or Domain in the aggregation end of Class with Property and Method).

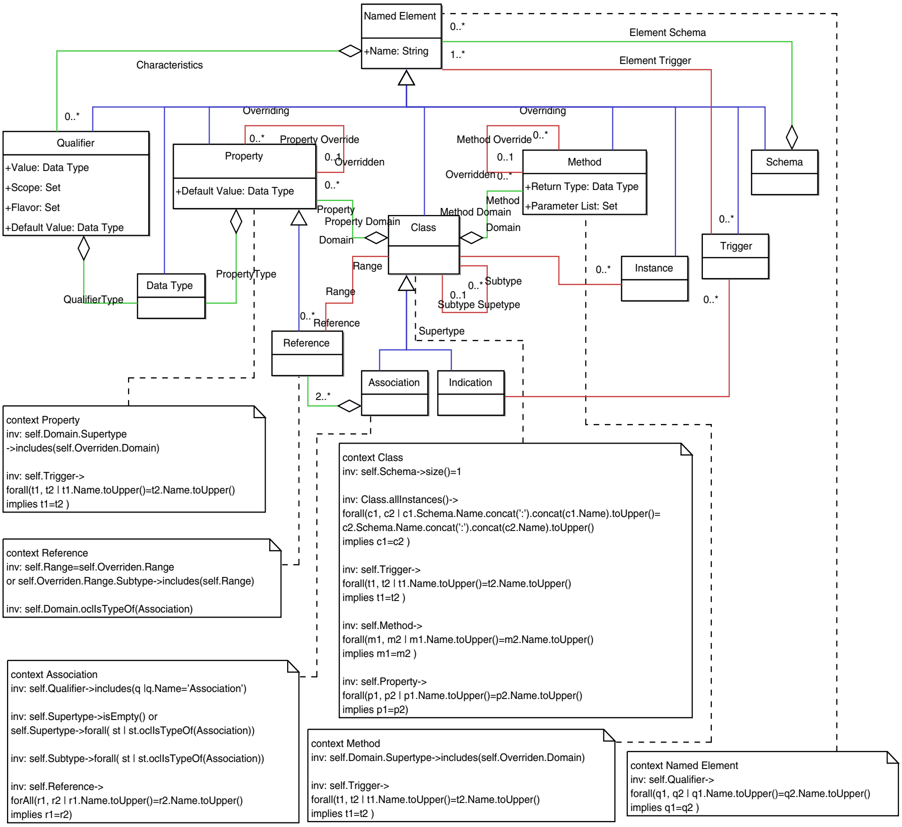
All those rules defined in English that could be formalized have been written in OCL, defining invariants inside the scope of each element. Other rules about the utility of each element were not formalized. Following lines present most important ones:

The rule that says that "A Class must belong to only one schema" has been specified as:

```
context Class
  inv: self.Schema->size()=1
```

The rule about overriding properties "The Domain of the overridden Property must be a supertype of the Domain of the overriding Property" has been defined as:

```
context Property
  inv: self.Domain.Supertype->includes(self.Overriden.Domain)
```



```
context Property
inv: self.Domain.Supertype
->includes(self.Overriden.Domain)

inv: self.Trigger->
forall(t1, t2 | t1.Name.toUpper()=t2.Name.toUpper()
implies t1=t2 )
```

```
context Reference
inv: self.Range=self.Overriden.Range
or self.Overriden.Range.Subtype->includes(self.Range)

inv: self.Domain.oclIsTypeOf(Association)
```

```
context Association
inv: self.Qualifier->includes(q |q.Name='Association')

inv: self.Supertype->isEmpty() or
self.Supertype->forall( st | st.oclIsTypeOf(Association))

inv: self.Subtype->forall( st | st.oclIsTypeOf(Association))

inv: self.Reference->
forAll(r1, r2 | r1.Name.toUpper()=r2.Name.toUpper()
implies r1=r2)
```

```
context Class
inv: self.Schema->size()=1

inv: Class.allInstances()->
forall(c1, c2 | c1.Schema.Name.concat(':').concat(c1.Name).toUpper()=
c2.Schema.Name.concat(':').concat(c2.Name).toUpper()
implies c1=c2 )

inv: self.Trigger->
forall(t1, t2 | t1.Name.toUpper()=t2.Name.toUpper()
implies t1=t2 )

inv: self.Method->
forall(m1, m2 | m1.Name.toUpper()=m2.Name.toUpper()
implies m1=m2 )

inv: self.Property->
forall(p1, p2 | p1.Name.toUpper()=p2.Name.toUpper()
implies p1=p2)
```

```
context Method
inv: self.Domain.Supertype->includes(self.Overriden.Domain)

inv: self.Trigger->
forall(t1, t2 | t1.Name.toUpper()=t2.Name.toUpper()
implies t1=t2 )
```

```
context Named Element
inv: self.Qualifier->
forall(q1, q2 | q1.Name.toUpper()=q2.Name.toUpper()
implies q1=q2 )
```

Fig. 3. CIM Metamodel formalized with OCL

A similar rule has been defined for methods: “The Domain of the overridden Method must be a superclass of the Domain of the overriding Method”.

```
context Method
  inv: self.Domain.Supertype->
    includes(self.Overridden.Domain)
```

There are some interesting rules about associations, such as “Associations are classes with an Association qualifier”, “An Association cannot inherit from a non-association Class”, or “Any subclass of an Association is an association”.

```
context Association
  inv: self.Supertype->isEmpty() or
    self.Supertype->forall( st |
      st.oclIsTypeOf(Association))
  inv: self.Qualifier->includes(q |
    q.Name='Association')
  inv: self.Subtype->forall( st |
       st.oclIsTypeOf(Association))
```

For references, rules like “The Class referenced by the Range association of an overriding Reference must be the same as, or a subtype of, the Class referenced by the Range associations of the Reference being overridden” or “The Domain of a Reference must be an Association” have been defined as follows:

```
context Reference
  inv: self.Range=
    self.Overridden.Range or
    self.Overriden.Range.Subtype->
      includes(self.Range)
  inv: self.Domain.oclIsTypeOf(Association)
```

There are some rules about qualifiers, but they reference elements that are not in the CIM metaschema class diagram, such as “A Qualifier Type (not shown in **Fig. 1**) is a Named Element and must be used to supply a type for a Qualifier (that is, a Qualifier must have a Qualifier Type). A Qualifier Type can be used to type zero or more Qualifiers” or “A Qualifier is a Named Element and has a Name, a Type (intrinsic data type), a Value of this type, a Scope, a Flavor and a default Value. The type of the Qualifier Value must agree with the type of the Qualifier Type”. They could be defined as:

```
context QualifierType
  inv: self.oclIsKindOf(NamedElement)
  inv: self.Qualifier->size()>=0
context Qualifier
  inv: self.QualifierType->size()=1
  inv: self.oclIsKindOf(NamedElement)
  inv: self.attributes()->
    includesAll(Set { 'Name', 'Type', 'Value', 'Scope',
'Flavor', 'Default-Value' })
  inv:
    self.Value.oclIsTypeOf(self.Type)
```

Other rules have been defined mainly to constraint that element names are case insensitive.

Redundant rules about multiplicity or specialization have not been included, because metaclasses associations already define them graphically in the diagram.

This formalization allows a compiler to check the defined information, by automatically processing the UML diagram with OCL rules, with just one possible interpretation, avoiding rules with multiple meanings.

5 Comparison with an Ontology Language

The approach given in this paper can be compared to other one presented in [4], where the use of the Web Ontology Language (OWL) [17] has been proposed to define management information. In this other work, the elements of this language have been studied, mapping them with management language constructions, and adding those facets not included in OWL that are common in management languages.

Both approaches could be valid depending on the application scope to enhance the semantic expressiveness of the information, as they provide different advantages and drawbacks, which can be taken into account when choosing the language more suitable for each case:

- CIM formalized version allows a smooth transition from the network management domain to the ontology domain. Moreover, with this approach it is not necessary to translate every CIM schema to other language, as they can be directly validated with defined OCL rules because the metamodel has been formalized. OCL can also be used in CIM to define constraints about the behavior of classes, methods and properties. Other UML artifacts different from OCL have also been used to describe behavior in [18]. Nevertheless, currently there are not tools to work with this information model from a semantic viewpoint. Another drawback is that this solution is CIM-centric: other management information defined in other language (for instance, SNMP MIBs) cannot directly profit from this approach.
- An ontology language such as OWL provides all the expressiveness of this kind of languages, because they are formalized. Also, there are many tools developed to use and validate it. In addition, other rule languages such as the Semantic Web Rule Language (SWRL) [14] can be used to define constraints about the behavior of that information. However, its main drawback is that all already defined management information has to be translated to OWL. Moreover, OWL does not allow the definition of class methods, so that part of the information can get lost.

These advantages and drawbacks can be compared in **Table 2**. As a conclusion, it can be said that CIM is better for current management tools, but OWL is better if ontology engines are used that analyze information to infer knowledge. The final decision can be based on the tools that are going to be used to handle the management information.

Table 2. Comparison of formalized CIM metaschema and OWL approaches

	Advantages	Drawbacks
Formalized CIM metaschema	• Smooth transition to the use of ontologies • CIM schemas are kept the same • OCL can also be used to define constraints for CIM schemas	• Semantic tools have to be developed • It only deals with CIM information
OWL	• Already formalized • Many developed tools • Definition of constraints with SWRL	• All management information has to be translated to OWL • Class methods cannot be defined in OWL

6 Conclusions

This paper has presented a proposal to formalize the CIM metaschema. For this, OCL has been used, rewriting the rules defined in natural language. With this, a compiler can load and interpret these rules to automatically check the semantics of defined information. This approach has also been compared with the use of an ontology language, obtaining that both solutions can be valid, providing each one some advantages and drawbacks.

There are some open issues. For instance, this formalization has been applied to CIM metaschema qualifiers, but not to qualifiers instances. This can be a problem, because these elements are used to extend the metaschema. However, in ontology languages every element is formalized. Thus, it would be necessary to carry out a formalization for every qualifier instance as performed above, specifying which invariants must be true for every element that have such qualifiers. This task is more complicated, because due to the qualifiers nature, metamodel and model levels get mixed.

Another future task is related to the measurement units. Currently, CIM only defines a list of values for the Units qualifier, but not their relationship. If they are formalized a property can be directly translated from a measurement unit to another. This formalization is useful if different classes are going to be compared. Then, for instance, two classes that measure the throughput of a channel can be mapped, even if one is in bits per second and the other in Megabits per second. Existing ontology libraries such as Ontolingua STANDARD-UNITS or DAML GNU Units can be leveraged with this purpose.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Education and Science under the project GESEMAN (TIC2002-00934).

References

1. R. Studer, V.R. Benjamins, and D. Fensel: Knowledge Engineering: Principles and Methods. Data & Knowledge Engineering. Vol. 25 (1998) 161-197.
2. J. E. López de Vergara, V. A. Villagrá, J. I. Asensio, J. Berrocal, Ontologies: Giving Semantics to Network Management Models. IEEE Network, Vol. 17, No. 3 (2003) 15-21.
3. E. Lavinal, T. Desprats, Y. Raynaud: A Conceptual Framework for Building CIM-Based Ontologies. In: Proc. of the Eighth IFIP/IEEE International Symposium on Integrated Network Management (IM'2003), Colorado Springs, Colorado, U.S.A., (2003)
4. 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.
5. G. Lanfranchi, P. Della Peruta, A. Perrone, D. Calvanese: Towards a new landscape of systems management in an autonomic computing environment. IBM Systems Journal, Vol. 42, No. 1 (2003) 119-128

6. S. Quirolgico, P. Assis, A. Westerinen, M. Baskey, E. Stokes: Toward a Formal Common Information Model Ontology. Lecture Notes in Computer Science, Vol. 3307, Springer Verlag (2004) 11-21
7. S. Bapat: Towards Richer Relationship Modeling Semantics. IEEE Journal on Selected Areas in Communications, Vol. 11, No. 9 (1993) 1373-1384
8. T. Zhang, PanosGavriil Tsigaridas: A Knowledge-based Model for Network Service Management. In Proceedings of the First IEEE Symposium Global Data Networking (December 1993)
9. Distributed Management Task Force, Inc.: Common Information Model Specification, Version 2.2. DMTF Standard DSP0004 (June 1999)
10. J. E. López de Vergara, V. A. Villagrá, J. Berrocal, J. I. Asensio, R. Pignaton: Semantic Management: Application of Ontologies for the Integration of Management Information Models. In: Proc. of the Eighth IFIP/IEEE International Symposium on Integrated Network Management (IM'2003), Colorado Springs, Colorado, U.S.A. (2003)
11. S. Cranefield, M. Purvis: UML as an Ontology Modelling Language. In Proc. of the Workshop on Intelligent Information Integration, Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm, Sweden (1999)
12. P. Kogut, S. Cranefield, L. Hart, K. Baclawski, M. Kokar, J. Smith: UML for Ontology Development. Knowledge Engineering Review Journal, Special Issue on Ontologies in Agent Systems, Vol. 17, Issue 1 (2002) 61-64
13. Object Management Group: Object Constraint Language Specification. OMG document formal/03-03-13 (March 2003)
14. 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)
15. Object Management Group: Unified Modeling Language (UML), version 1.5. OMG document formal/03-03-01 (March 2003)
16. Distributed Management Task Force, Inc.: Common Information Model (CIM) Infrastructure Specification, Version 2.3 Preliminary. DMTF Standard DSP0004 (October 2004)
17. D. L. McGuinness, F. van Harmelen: OWL Web Ontology Language Overview. W3C Recommendation (10 February 2004)
18. M. Sibilla, A. Barros de Sales, J. Broisin, P. Vidal, F. Jocteur-Monrozier: Behaviour modelling: a contribution to CIM. DMTF Academic Alliance Paper (2004)

Ontology-Based Integration of Management Behaviour and Information Definitions Using SWRL and OWL

Antonio Guerrero[1], Víctor A. Villagrá[1], Jorge E. López de Vergara[2], 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, berrocal@dit.upm.es

Abstract. Current network management architectures are using different models to define management information objects. These definitions actually also include, in a non-formal way, the definition of some behaviour information that a manager should accomplish related to the managed objects. So, a manager is not able to make an automatic processing of this behaviour information. Prior research work proposed the use of formal ontology languages, such as OWL, as a way to make a semantic integration of different management information definitions. This paper goes further proposing a formal definition of the different management behaviour specifications integrated with the management information definitions. Thus, usual behaviour definitions included implicitly in the management information definitions and explicitly in policy definitions can be expressed formally, and included with the information definitions. This paper focuses on the definition of behaviour rules in management information with SWRL, a rule language defined to complement OWL functionality.

1 Introduction

The heterogeneity of the resources found in telecommunications networks and services has led to the definition of several integrated management architectures, which are intended to manage a heterogeneous environment by using a common mechanism for accessing the management information. Initially, two frameworks emerged: the OSI management architecture and the Internet management architecture. While the Internet management architecture has gained widespread acceptance, OSI management is still used in some cases, especially in telecommunications networks managed with the TMN architecture.

In addition, by the end of the 1990's, a new framework, which makes use of the web protocols, came onto scene with the same goal of achieving integrated management: the WBEM architecture (Web Based Enterprise Management). Nevertheless, this new architecture did not exclude the previous ones, including some interoperability with different integrated network management architectures.

In this environment with multiple integrated management frameworks, the initial problem that those architectures tried to solve arises once again, since one manager

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 12 – 23, 2005.

has to interact with different resources using several different mechanisms. Even so, the manager cannot achieve a truly integrated management of the environment: since it cannot know about the semantics of the managed resources defined in the different models, it cannot apply common policies to manage those resources, which are in fact the same. Policies should be generic and independent of the model in which the resources are defined.

In order to solve this problem, a proposal of the so-called Ontology-based Semantic Management is included in [1]. This semantic management allows a 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). Another advantage of this approach was also pointed out in that proposal: it allows the integration, in that same unified management information model, of the definition of behavioural characteristics of the manager and of the resources. These behavioural aspects are usually found as comments inside the definitions, or explicitly declared outside of the definitions of the managed resources. With this approach, all the definitions (information and behaviour) are integrated, so they can be jointly processed automatically.

This paper proposes a starting point for this integration: beginning with the unified information model, defined in an ontology language such as OWL, it proposes inclusion in this model of the behaviour definitions, expressing them by means of the Semantic Web Rule Language, SWRL. For this purpose, next section shows the semantic management architecture and the SWRL language. This language will allow definition of the behaviour as part of the definitions of the management information, as will be shown. Also, the different types of behaviour are explained: implicit restrictions, explicit behaviour of the manager, and behaviour of the managed elements. Example definitions in SWRL are included for each one of these types. Lastly, the most relevant concepts are summarized.

2 Semantic Management

The global architecture proposed in [2] is based on a manager that works and reasons with a unique information management model, represented by means of ontologies. This system manages elements from different domains (SNMP, CIM, etc.) from a common and neutral perspective. **Fig. 1** shows the proposed architecture of the semantic manager.

The main goal is the usage by the manager of only one information model, but, in many cases, the different network resources are defined in different management domains (SNMP MIBs, CIM schemas, etc.), so they have to be accessed by using the predefined protocols for those domains. Therefore, it is necessary to translate and integrate those definitions into one unified specification, taking into account the semantics of the initial definitions. In other words, the same resource, defined twice in two different models, will have one single representation in the unified model, and will have two translations to the original models. So, the goal is not just to make a syntactic translation of the definitions, but also about their integration from a semantic viewpoint.

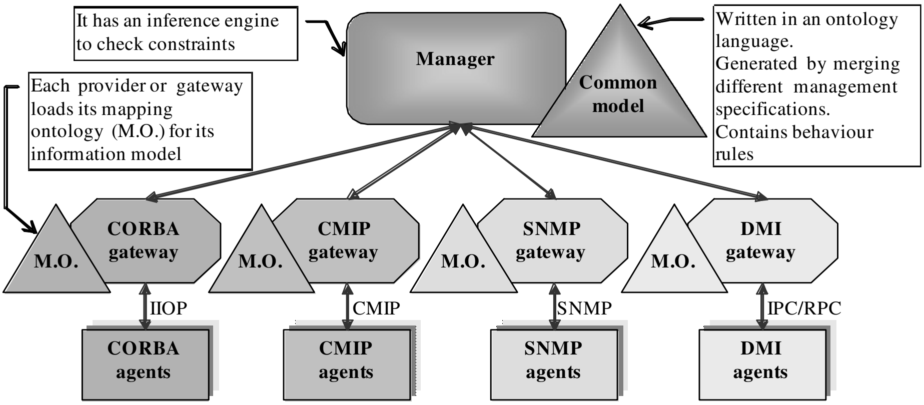

It has an inference engine to check constraints
Manager
Common model
Written in an ontology language.
Generated by merging different management specifications.
Contains behaviour rules
Each provider or gateway loads its mapping ontology (M.O.) for its information model
M.O.
CORBA gateway
M.O.
CMIP gateway
M.O.
SNMP gateway
M.O.
DMI gateway
IIOP
CMIP
SNMP
IPC/RPC
CORBA agents
CMIP agents
SNMP agents
DMI agents

Fig. 1. Proposed architecture for the semantic manager [2]

The resulting mappings from the original definitions to the ontology specification are then used by the so-called "providers" or gateways in **Fig. 1**. They are responsible for translating the information of the network elements in the unified model back into the information in their original management models.

Using an ontology language for defining the management information has additional advantages, such as the possibility to use existing tools in order to work and to reason with the ontologies (for example inference engines used in artificial intelligence).

Another advantage, already mentioned in the previous section, is that a management ontology allows the integration of rules defining the expected behaviour of the management information. In this way, the behaviour definitions that are usually implicitly included in the management information definitions (declared in natural language, or tacitly supposed), can now be formally expressed as an integrated part of the management information definitions, and in their same language (ontology language). This approach implies that the behaviour definitions are now formally expressed in the same definition language, and that they can be interpreted and validated or enforced by the same semantic manager, in order to work and reason with them. All definitions, information management (MIBs) and behaviour rules (policies), are now integrated in the same network management ontology. One of the main advantages of this approach is that it will permit generic managers to behave depending on definitions, instead of having embedded encoded behaviour.

OWL [3] is proposed [4] as the ontology language for the definitions, a general purpose ontology language defined for the Semantic Web that contains all the necessary constructors to formally describe most of the information management definitions: classes and properties, with hierarchies, and range and domain restrictions. SWRL [5] extends the set of OWL axioms in order to include conditional rules (Horn clauses), of the form *if… then …*

Axioms and rules can be used in this management framework in order to:

1. Further constrain or define more precisely the behaviour of the OWL management information. This will guarantee the correct use and implementation of the management information.

2. Formally define the behaviour of the manager. This allows the declaration of the manager actions when certain conditions are met on the managed elements.
3. Formally define the behaviour of the managed objects. This allows the definition of what the managed resources should do upon certain events or conditions.

The purpose of this paper is to integrate the definition of the management information (expressed in OWL) with the definition of the management behaviour (expressed in SWRL). The next section explains briefly the SWRL language, used to define rules for OWL ontologies.

3 SWRL: Definition of Rules for OWL Ontologies

An OWL ontology contains a sequence of axioms and facts. It includes several types of axiom, such as subclass axioms, equivalent Class axioms and property constraints. SWRL proposes to extend these with rule axioms.

A rule axiom consists of an *antecedent* (body) and a *consequent* (head), each of which consists of a (possibly empty) set of atoms.

In the “human-readable” syntax of SWRL, a rule has the form:

```
antecedent ⇒ consequent
```

Informally, a rule may be read as meaning that if the antecedent holds (is "true"), then the consequent must also hold. Using this syntax, a rule asserting that the composition of parent and brother properties implies the uncle property would be written:

```
Person(?x) ∧ isFather(?x,?y) ∧ isBrother(?y,?z) ⇒
isUncle(?x,?z)
```

An SWRL rule has therefore the form of an implication relationship between the head and the body. The SWRL specification [5] provides an abstract syntax that extends the OWL abstract syntax described in [6] to include this relationship in the ontology language. The XML labels used for defining rules include:

- <ruleml:imp>: it is the element that relates the body of the rule with the head (the label of the relationship).
- <ruleml:_body>: it is the element that contains the atoms that form body of the rule.
- <ruleml:_head>: it is the element that contains the atoms that form the head of the rule.
- <ruleml:var>: it allows the definition of the variables used in the evaluation of rules.
- <swrlx:individualPropertyAtom>: it allows the definition of atoms that refer to specific properties. It is also possible to define atoms that refer to classes, data ranges, valued properties, or typed functions such as mathematical, dates and strings.

As is shown, many of these labels are not defined inside the SWRL namespace, but in RuleML’s [7], a rule language previously defined that has been taken as the base for SWRL definition. SWRL mainly brings the definition of atoms and the integration of these rules into an ontology written in OWL.

4 Definition of Management Behaviour in OWL+SWRL

4.1 Types of Management Behaviour

In order to define management behaviour let us first classify which types of behaviour can exist. Concretely, three types of management behaviour have been identified:

1. Implicit constraints and rules about the behaviour of the modelled objects in the MIBs and CIM schemas
2. Explicit behaviour of the manager, in the traditional manager-agents architecture, which defines how the manager should behave upon analysing the information obtained from the agents
3. Explicit defined policies to specify the dynamic behaviour or dynamic configuration of the managed resources (Policy-Based Management [8])

In this semantic management framework, policies can be defined in the same management language – OWL+SWRL – in which the objects of the management information base (MIB) are defined, with the advantage of working with a unified model.

Going back to the management architecture proposed in Fig. 1, the constraints, rules, and policy definitions would be stored in the rule information base – part of the common model, whose purpose is verifying the integrity of the information, and automating the control of the managed elements.

The following sections focus on each of the identified behaviour types.

4.2 Implicit Restrictions on the Management Information

This kind of rules refers to restrictions in the data type, cardinality, or access. They are typical restrictions upon the properties and classes of the managed objects. In this case SWRL rules complement the existing mechanisms in OWL to form the management information definitions, in the sense that they allow expressing more complex restrictions: values that depend on the values of other variables, relationships among objects, state-machine behaviour of the values, etc.

In a general way, SWRL allows the representation of behaviour restrictions that can be expressed in a natural language as conditional clauses (*if… then …*). This includes, for example, values that depend on other values, state-machine behaviour, temporal behaviour, or complex types such as composed functions. The following subsections present some examples of these kinds of behaviour found in SNMP MIBs, and how SWRL is used to formally define these restrictions. In many cases, the definition of the restrictions will require the creation of new classes and properties that would extend the management ontology.

Example 1

This first example shows how to specify in SWRL that the value of one variable depends on the value of another variable. It is based on a definition from SNMP's MIB II, which restricts the value of the mask for route entries, in the routing table:

```
ipRouteMask OBJECT-TYPE
          SYNTAX IpAddress
          ACCESS  read-write
          STATUS  mandatory
          DESCRIPTION
```

```
                "… If the value of the ipRouteDest is 0.0.0.0 (a
default route), then the mask value is also 0.0.0.0 …"
            ::= { ipRouteEntry 11 }
```

This implicit restriction is expressed in natural language in the DESCRIPTION clause, but it is not formally defined in SMIv2, so it cannot be automatically implemented by a manager when the MIB is compiled.

If theMIB II has been mapped and integrated in the management information base in OWL, the SWRL rule to define this example restriction would be the following:

```
ipRouteEntry(IR?) ∧ swrlb:equal (ipRouteDest(IR?), "0.0.0.0")
⇒ swrlb:equal(ipRouteMask(IR?), "0.0.0.0")
```

where ipRouteDest and ipRouteMask are properties of the class ipRouteEntry.

It is interesting to notice that this kind of restriction, hereby expressed in a simple manner, is not currently supported by existing management definition languages. In fact, the IETF’s SMIng Working Group proposed the following objective for the next generation of the SMI definition language, stated in [9]: “*SMIng should provide mechanisms to formally specify constraints between values of multiple attributes*”, but its implementation was not accomplished: “*This objective as is has been rejected as too general, and therefore virtually impossible to implement*”.

On the other hand, further reinforcing the idea under study, it so happens that OWL without SWRL can neither express this constraint: a restriction on a property (owl:restriction clause) cannot be made to depend on the value of another property of the same object.

Example 2

This second example, also obtained from SNMP’s MIB II, can be used to show how to define state-machine behaviour with SWRL. It is based on the column of the TCP connection table which shows the state of each connection.

```
tcpConnState OBJECT-TYPE
 SYNTAX INTEGER {
    closed(1),
    listen(2),
    synSent(3),
    synReceived(4),
    established(5),
    finWait1(6),
    finWait2(7),
    closeWait(8),
    lastAck(9),
    closing(10),
    timeWait(11),
    deleteTCB(12)
  }
 ACCESS  read-write
 STATUS  mandatory
 DESCRIPTION "…"
::= { tcpConnEntry 1 }
```

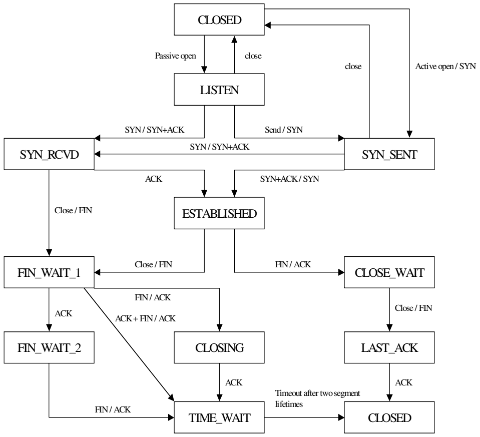


Fig. 2. TCP state machine and definition

Since this parameter has a read/write access, this example is intended to define the state-machine diagram shown also in **Fig. 2**, so that a manager could check that all connections comply with the state machine.

For this purpose, in the management ontology we define a new class of objects called tcpConnEntry, with a property that indicates the actual state of the connection, tcpConnEntry, and auxiliary properties tcpConnPreviousState and tcpConnNextState, which indicate the list of possible before and after states. The following is an example rule that could be used to define the state-machine:

```
tcpConnEntry(cx?) ∧ swrlb:equal(tcpConnState(cx?),
"fin_wait_1") ⇒ swrlb:member(tcpConnNextState(cx?),
nextStatesForFin_Wait_1_List)
```

where nextStatesForFin_Wait_1_List would be a list (rdf:list) with the values “closing”, “time_wait” and “fin_wait_2”.

Example 3

The third example is a case of defining a temporal constraint, also described in natural language in SNMP MIB II. It is a constraint upon the value of the column of the interfaces table which indicates the latest time at which the interface changed. The constraint definition for the property ifLastChange that can be represented in SWRL is marked in bold letters:

```
ifLastChange OBJECT-TYPE
          SYNTAX TimeTicks
          ACCESS read-only
          STATUS mandatory
          DESCRIPTION
            "The value of sysUpTime at the time the interface
entered its current operational state.  If the current state
was entered prior to the last re-initialization of the local
network management subsystem, then this object contains a
zero value."
          ::= { ifEntry 9 }
```

To implement this constraint it is also needed to use auxiliary classes and properties. In this case, the temporal comparison can be made making use of the property sysUpTime of the class system.

```
system(?x) ∧ ifEntry(?y) ∧ isInterfaceOf(?y, ?x) ⇒
swrlb:lessThan(ifLastChange(y?), sysUpTime(?x))
```

4.3 Explicit Behaviour of the Manager

The behaviour of the manager when certain conditions are met on the network or the managed systems, can also be specified by means of conditional rules *if condition then action*, which become the following in SWRL

```
condition ⇒ action
```

or, making use of the broader definition of SWRL:

```
condition set ⇒ action set
```

Actions

As a management information definition language, OWL lacks the explicit ability to define operations or methods on the managed objects that are typical in other management information languages such as CIM or GDMO. In order to solve this problem, actions in OWL can be represented by means of the OWL-Services ontology, OWL-S [10], which allows an easy integration with the rest of definitions. One of the classes that this ontology defines, among many others, is the Process class that can be used for this purpose. In this way, "executing an action" can be interpreted as "calling a service" that executes that action, and so it is possible to define that action as an OWL-S Process, and then instantiate an object of the class Perform, with such process as its argument:

```
Perform(MyProcess)
```

The class Perform is an auxiliary class used in OWL-S to represent the execution of atomic processes inside a composite process, e.g. Perform(Reset), Perform (SendAlarm(…)), Perform(SetIPRoute(…)), etc.

These processes can be either newly defined actions or the result of integrating existing operations or methods of the merged information models (CIM methods, SNMP "set" operations, etc.)

In the proposed architecture in Fig. 1, which follows the traditional manager-agent paradigm, the manager would invoke the service calls that the providers would offer, and these providers would in turn execute the requested operations upon the managed elements, in their native languages and protocols.

The following example shows this type of definition of the behaviour of the manager upon the existence of certain conditions on the network:

Example 4

This example makes use of the class CIM_SystemDevice from the CIM schema, with two instances that are port devices. With this SWRL rule, the manager will activate the second port if the first port is not working, i.e.:

```
If LogicalPort #1 is "Operatively Down", then enable
LogicalPort #2
```

In SWRL:

```
CIM_SystemDevice(LP1?) ∧ swrlb:equal(deviceName(LP1?),
"Lport1") ∧ CIM_SystemDevice(LP2?) ∧
swrlb:equal(deviceName(LP2?), "Lport2")  ∧
swrlb:equal(StatusInfo(LP1?), "OPERATIVELY_DOWN") ⇒
Perform(SetAdminAvailability(LP2?, "ENABLE"))
```

In this case, the rule is applied upon certain instances of a class, not upon all elements of the class.

In the same way as it occurred with implicit restrictions, this type of explicit definitions could neither be expressed in OWL without SWRL, as it is a restriction upon the values of a property. Also, there is a clear difference in this example with the example #1 in the previous section, because in this case the related values are the values of the same property, from two different objects of the same class, instead of values from different properties of the same object.

4.4 Explicit Behaviour of the Managed Elements: Application to Policy-Based Management

If the rules being defined in the information model are not rules for the behaviour of the manager, but rules that define the behaviour of the managed elements, then we need an architecture that supports the distribution of the rules or policies to those managed elements. This type of architectures is defined in the PBM framework (Policy-Based Management) or policy-based networking (PBN). This work references the following elements of the PBM architecture, proposed by the IETF/DMTF [8]:

- PEP Devices (Policy Enforcement Point): those elements that can apply or execute the policies.
- PDP Devices (Policy Decision Point): they act as proxy between the PEPs and the policy repository, being responsible for interpreting the policies from the repository and indicating the corresponding actions to the PEPs.
- Policy Repository: it stores the defined policies that will be distributed to the PDPs.

A policy-based architecture presents the following key characteristics:

- Centralization: definition of behaviour is made in a single point and can be massively distributed throughout the network, instead of being defined and applied individually for each element
- Abstraction Levels: policies can be defined at different level: high level policies (i.e. business rules), intermediate level (i.e. service level rules), and lower level (i.e. policies applied by the network elements). It might be necessary to translate policies among levels, in order to convert high level rules into the lower level policies that will be applied by the network elements. For this task it will be useful to define models (business models, service models, network models). Existing policy definition languages can be oriented to policy definition at determinate levels. For example, PONDER and RBAC would be classified as high-level policy definition languages, since the information model is closer to natural language and human thinking. On the other hand, PCIM and COPS-PR PIB allow the definition of lower level policies, closer to the management languages used by the network elements.

In this section, we will attempt a first approach to the possibility of using ontologies in OWL+SWRL as a definition language for those kinds of high and low level policies, through the following examples.

Example 5: High Level Policy in PONDER

This example matches in SWRL a policy previously defined in PONDER, extracted from [11]:

```
type rel ReportingT (ProjectManagerT pm, SecretaryT secr) {
          inst oblig reportWeekly {
            on timer.day ("monday") ;
            subject secr ;
            target pm ;
            do mailReport() ;
          }
          // . . . other policies
}
```

The obligation policy reportWeekly specifies that the subject of the SecretaryT role must mail a report to the subject of the ProjectManagerT role every Monday. In SWRL it could be expressed like this

```
ProjectManagerT(pm?) ∧ SecretaryT(secr?) ∧ Timerday(t?) ∧
swrlb:equal (t?, "monday") ⇒  Perform(mailReport(secr?, pm?))
```

In a similar way, other kinds of policies could be mapped to SWRL: authorization policies, obligation policies, filtering policies, etc.

Example 6: Low Level Policy
Next, an SWRL definition will be shown of a low level policy that could be executed by a network element.

The following rule would set to “0100” the ToS (Type Of Service) field of each IP packet whose destination address belonged to the specified address range.

```
IPpacket(ippacket?) ∧ InsideIPRange(DestAddress(ippacket?),
"192.168.1.0 - 192.168.1.255") ⇒
Perform(SetTOSlabel(ippacket?, "0100"))
```

While this example demonstrates the possibility of defining these kinds of lower level policies, it might be too simple and not very useful in a real case. A better approach to integrate these kind of policies into the unified management ontology would be to attempt the mapping of the existing PIBs (Policy Information Base), or mapping the existing policy definition languages that the network elements or PDPs understand. Two kinds of necessary mappings can be identified: on one side, the mapping of the conditions that trigger the policies, or condition sets; on the other side, the mapping of the operations to be performed, or action sets. If the same kinds of policy are defined in different definition languages, the integration approach could use the M&M (Merge and Map) method [2], proposed for the integration of management information definitions coming from different management models to a common ontology. The application of this method would in turn allow a reverse translation, from the policies represented in the OWL+SWRL ontology, to the native management languages that the PDPs would understand.

At this point it is necessary to notice that the policies represented in the examples hereby included (such as *“<condition set> then do <action list>”*) are policies without events. There are no explicit events to trigger the evaluation of the policies, but the agents must do this task whenever there is an implicit event, as would be starting a new session, or periodically. This would be the case of the PCIM definition language, but not of COPS and PONDER, since these languages allow the definition of policies with this kind of event-condition-action.

Policy-based management also deals with translation of policies among different abstraction levels, so policies defined at higher levels (i.e. business level and service level) can be translated down to lower level policies (i.e. network and system levels). In the semantic management approach, policy definitions at all levels would be defined in the same definition language, OWL+SWRL, which could facilitate the process of translating among levels. Nevertheless, this is currently out of the scope of the present work.

5 Conclusions

As it has been shown in previous research work [1,2,4], ontology languages such as OWL include the constructions necessary to define the typical aspects of the network management information that can be found in other management definition languages such as SMI-SNMP, MOF-CIM, etc., so it is possible to make a mapping and merging process which integrates definitions in different languages from a semantic point of view.

In this semantic management framework, the present work has shown how SWRL:

1. adds expressiveness power for defining constraints and rules for the proposed network management information ontology in OWL. This means that more complex constraints and policies can be formally expressed in the OWL management information model, therefore enriching and empowering the overall semantic framework for network management
2. allows explicitly defining the behaviour of the manager, and of managed objects, in the same language – OWL+SWRL – that is used for the definitions in the management information base. This includes:
   - Actions that will be performed by the manager upon certain conditions on the network objects or the manager itself
   - Actions that will be performed by the network elements upon the existence of certain conditions (policies that define the behaviour of these managed elements)

In this semantic management framework, some management architecture elements have yet to be defined, including events, translation and distribution of policies, and other elements from policy-based networking.

Another direction for future work is the approach to be followed for accomplishing the formal definition of the existing restrictions and rules implicitly or explicitly defined. For this work it would be useful to develop tools that propose these restrictions using heuristic searches of constraints described in the “description fields” (e.g. strings including “if*then”, “have to” or “must” will have a high probability of being part of these natural language constraint definitions), and their final representation in SWRL making use of the existing OWL definitions.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Education and Science under the project GESEMAN (TIC2002-00934).

References

1. J. E. López de Vergara, V. A. Villagrá, J. I. Asensio, J. Berrocal: Ontologies: Giving Semantics to Network Management Models, IEEE Network, Vol. 17, No. 3 (2003) 15-21
2. J. E. López de Vergara, V. A. Villagrá, J. Berrocal: Benefits of Using Ontologies in the Management of High Speed Networks. In *High Speed Networks and Multimedia Communications – Proceedings 7th IEEE International Conference, HSNMC 2004.* Toulouse, France, June 2004. LNCS 3079: 1007-1018, Springer-Verlag (2004)
3. M. K. Smith, C. Welty, D. L. McGuinness: OWL Web Ontology Language Guide, W3C Recommendation (10 February 2004)
4. 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

5. 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)
6. P. F. Patel-Schneider, P. Hayes, I. Horrocks: OWL Web Ontology Language Semantics and Abstract Syntax", W3C Recommendation (10 February 2004)
7. Rule Markup Initiative: http://www.ruleml.org/
8. 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)
9. C. Elliott, D. Harrington, J. Jason, J. Schoenwaelder, F. Strauss, W. Weiss: SMIng Objectives, IETF Request For Comments 3216 (2001)
10. D. Martin, editor: OWL-S: Semantic Markup for Web Services, W3C Member Submission (22 November 2004)
11. N. Damianou, N. Dulay, E. Lupu, M. Sloman: The PONDER Policy Specification Language. In *Proc. International Workshop of Policies for Distributed Systems and Networks (Policy 2001)*. Bristol, UK, January 2001. LNCS 1995: 18-39, Springer-Verlag (2001)

On the Impact of Management on the Performance of a Managed System: A JMX-Based Management Case Study

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. Studying the performance of a distributed system without taking care on the impact of its management system will falsify the understanding of its overall performance, especially its productivity. We propose a metric called *MIM (Management Impact Metric)* to evaluate this impact by varying one or several impact factors related to the management system within a management strategy of the managed system. We show the accuracy and interest of our metric on a managed J2EE application server that uses a management architecture based on the JMX standard.

Keywords: Managed systems performance, productivity, manageability.

1 Introduction

The essence of modern networks and services (home gateways, sensor networks, application servers, grids) lies in the optimal utilization of resources within a dynamic and large working environment. A key component required in this respect is the management framework that monitors these systems and orchestrates their activities to improve and maintain their performance. The goal of management is to ensure that the managed systems operate with the efficiency and effectiveness predefined in the quality of service parameters. Since current network management architectures are often integrated in the service activities, it is essential to be able to know the overhead of these management activities and their impact on the overall performance of the managed systems. A basic question we are trying to answer here is: *How do management activities impact the overall managed system performance ?* and *How can we minimize this impact ?*

Management architectures and their associated activities are becoming very complex and diverse. Over the last 20 years, new and enhanced management architectures appeared, varying from OSI and SNMP(v1,v2,v3) to Web Sservices-based management including Java specific approaches like the JMX standard which became very popular. Such management architectures have the following characteristics:(1) their activities are essential to manage the system;(2) they offers a set of management strategies that operate differently on the managed system (e.g., polling vs notification); (3) they can severely impair the performance of the user's work (referred to as **productivity**) if their overhead cost per management strategy is not well defined and studied. In the literature, many studies [1,2] evaluate the performance of these management architectures,

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 24–35, 2005.

and they focus especially on comparing their performance. However, the question of how management activities impacts the performance of managed systems has not been studied so far. The variability of performance captures the impact of management on the performance of a managed system.

A metric that quantifies this impact should be defined. It must put in relation the performance of management and managed systems. To address this issue, we propose an analytical model of this impact that combines the performance of the management and the managed systems over varying management profiles and under an impact factor.

The paper is structured as follows: Section 2 reviews related works. Section 3 presents the set of the impact modifiers and factors of a management system that impact the performance of a managed system. Section 4 introduces the analytical model of our impact function. Section 5 presents an example of using the impact metric on a JMX based management activity of the JBoss application server. Section 6 contains a brief summary of this contribution as well as an outlook.

2 Related Work

Several papers separate investigations on the performance of distributed systems and the performance of management systems. Woodside and al. [3] define the performance referred as **productivity** of a system as the relation between the rate of providing valuable services, the quality of those services and the cost of providing those services. Another definition is proposed in [4], where performance is viewed as the response time, seen by a user under normal working conditions, coupled with the cost of the system - hardware requirements - per user. We will adopt the performance metric proposed for distributed systems to assess the performance of the managed and management systems. In fact, we will use the same productivity definition as defined in [3] to quantify the performance of the managed system under an impact factor.

Several studies are related to the performance evaluation of specific management systems. The focus of most of these studies has been to model the performance of management architectures and their associated cost. Their performance models quantify response time of agents [1,5], the volume of management traffic [2] and resources usage [6,7]. Nevertheless, all performance studies related to management architectures that take as elementary performance metrics : response time, management requests rate and resource usage will benefit from our management impact metric. Our proposed metric is based on the efficiency [8] of the managed system. This function is defined as the useful work of the managed system divided by the total work (productivity+manageability). By continuously computing the impact metric, a management system will regulate its activities to minimize its impact or adapt the management profile (strategy) parameters within the managed system. This metric provides an auto-tuning criterion for the management system [9], which allows the managed system to be more self-managing and more efficient.

3 Management Profiles

Despite the wide variety of management technologies and products, most management system infrastructures fall into an architecture pattern referred to as Manager-Agent.

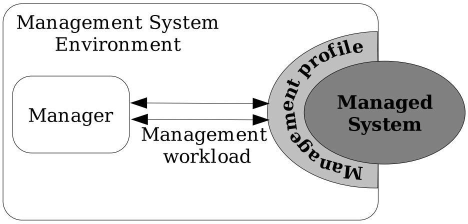


Fig. 1. The Management system components

Table 1. The management profile parameters and impact factors

Management profile parameters (Impact modifiers)	Management Impact factors
Management approaches: - Client-server - Hierarchical static - Weak mobility - Strong mobility	- Number of management requests - Number of notifications - Number of management objects within the managed system - Management requests mixes : * Read requests * Write requests
Management enabling technologies: SNMP, RMON/RMON2, CMIP, WBEM, PBN, Mobile agents, DCOM, CORBA, JMX	
Management tasks: - Monitoring - Tracking - Control	
Management operation models: - Polling - Notification	
Agent deployment models: - Deamon - Component - Driver	
Management instrumentation models: - External - Internal	

There are three basic components in this architecture: managed system, agent and management applications. The management application is responsible for providing the infrastructure and user interfaces to manage a system and it is conducted by a management profile or strategy that defines manageability tasks and patterns (see figure 1.)

> *Definition:* A management profile is a quantitative characterization of how a system is managed. The profile summarizes key interaction parameters between the management system environment and the managed system.

A management profile covers the most important parameters related to the management system and its interaction with the managed one. It is important to identify those parameters, which, if varied, will change the management profile within the managed system. Parameters that are changed from a management profile to another are called *impact modifiers*. These modifiers, if varied, will have a significant impact on the performance of the managed system. The impact modifiers might improve, maintain or degrade a given managed system performance. Examples of impact modifiers are the agent deployment patterns within the managed system, management tasks and their operation patterns (polling-driven or events-driven), the design patterns of management objects within the managed system. The management profile is controlled by the management workload that includes management requests. The management workload characterization parameters represents the set of *impact factors*.The impact factors denote a set of impact variables, determined by the management profile within the managed system. The impact metric is analysed by varying the impact factor within a management profile. Table 1 displays a non exhaustive list of management profiles and factors that affect the performance of a managed system. Within these profiles, three parameters are chosen and studied in more details in the paper. For each parameter, the intuitive impact on the performance of the managed system is listed.

3.1 Management Tasks

As defined in [10], management system tasks are the following:

- Monitoring: the ability to capture runtime and historical events from a particular component. This task is continuous over the execution time of the managed system and it is executed concurrently with users on each monitoring cycle.
- Tracking: the ability to observe aspects of a single unit of work or thread of execution across multiple components (e.g., tracking messages from senders to receivers). This task is executed less frequently than the monitoring on a period from the execution time of multiple components within the managed system.
- Control: the ability to alter the runtime behavior of a managed component (e.g., changing the logging level of an application). The execution of this task can result from the first two ones when problems detected by monitoring or tracking need to be resolved by controlling the managed system. This task is executed on a precise period from the execution time of the managed system.

Thus, it is easy to see that the monitoring task will introduce a periodically impact on the performance of a managed system. However, the control and tracking tasks do not permanently affect the performance of the managed system (An example of a management profile for the JBoss server is given in section 5).

3.2 Management Agents Deployment Models

The way in which the management agent is deployed within the managed system is an impoortant profile parameter. In [11], the authors identify 3 management agent deployment models: daemon, component, and driver. In the daemon model, the agent owns its own process separate from the application. In this case, the managed component and the agent do not share the same resources and may running on two different hosts. The sole overhead introduced by the agent on the managed component is the communication cost to retrieve management data from the managed resource. In the component model, the agent runs in the process owned by the application and they share the same resources. Hence, the overhead of the managers interacting with the agent is added to the resources used by the managed application. The driver deployment model is similar to the component model. Rather than a component, the agent become the core of the system. In this case, all manageability work is executed concurrently with the users work.

3.3 Management Instrumentation Patterns

This management profile parameter specifies the way in which the management object (e.g.,MBeans for JMX) retrieves the management data from the managed resource. Two patterns are identified [11]: internal instrumentation and external instrumentation. In the internal instrumentation the management object is part of the managed resource and management tasks are executed directly on it. External instrumentation is defined and executed outside the managed resource. From these definitions, internal instrumentation might affect more significantly the performance of the managed resource rather than the external one.

We can see clearly that the choice of a management profile or a strategy rather than another will modify the potential impact of the management system on the performance of the managed system.

4 The Impact Function

The impact function is designed to capture the performance variability of a managed system under a management profile at a given impact factor value. We named the impact metric MIM as *Management Impact Metric*. $MIM(k)$ is a function that maps the impact factor k to a value within the closed interval [0, 1]. It indicate whether a performance degradation has occurred, and includes an indication of the degree of that degradation. The $MIM(k)$ function distinguishes between an unacceptable impact of a management system (for which $MIM(k)$ is close to 1) and an acceptable impact (for which $MIM(k)$ is close to 0).

Instead of productivity, which is the performance metric (production work) related to the managed system, we name the performance of the management system as **manageability** (management work) [10]. We denote $F(k)$ as the productivity of the managed distributed system and $G(k)$ as the manageability of the management system at an impact factor k. Hence, the efficiency of the managed system at the impact factor k is given by:

$$E(k) = \frac{F(k)}{F(k)+G(k)} \tag{1}$$

We adopt the productivity F(k) of the managed system or the manageability G(k) of the management system defined in [3] as follows:

$$F(k) = \lambda_1(k).\frac{f(k)}{C(k)}, G(k) = \lambda_2(k).\frac{g(k)}{C(k)} \tag{2}$$

Where $\lambda_1(k)$, $\lambda_2(k)$ are respectively the users work throughput in responses/sec of the managed system and the management throughput in responses/sec of the management system at an impact factor k. The function $f(k)$, respectively $g(k)$, is an average value of each response calculated from its quality of service at the impact factor k. The value function $f(k)$ is determined by evaluation of the performance of a system (managed and management ones), and may be a function of any appropriate system measure including delay measures (mean, variance or jitter, probability of delay exceeding a threshold). In this work we will consider only the mean response time $T(k)$ at the impact factor k, normalized to a target value $\bar{T}$ (response time quality of service), in the following value function [12]:

$$f(k) = \frac{1}{1+(\frac{T(k)}{\bar{T}})} \tag{3}$$

The target value $\bar{T}$ is an optional upper bound for the delay that can be specified for an impact state to be acceptable. If we do not specify the delay target value, the value function $f(k)$ (respectively $g(k)$) will be the following [3]: $f(k) = \frac{1}{T(k)}$. In this case the productivity is given by:

$$F(k) = \frac{\lambda_1(k)}{T_1(k)}.C(k), G(k) = \frac{\lambda_2(k)}{T_2(k)}.C(k) \tag{4}$$

$C(k)$ is the cost function at the impact factor k, expressed as the running cost per second to be uniform with λ_1 (respectively λ_2). The cost may be a function of any appropriate weighted sum of resources utilization metrics such as cpu, memory and network. The weight coefficients imply their importance on the managed system. Then $C(k) = a.CPU(k) + b.Memory(k) + c.Network(k)$, where a, b and c are the weights of the resources consumed either by the managed system or the management one. The function $E(k)$ denote the efficiency of the managed system associated with an impact state k, under a management profile characterized by its manageability $G(k)$. The impact function $MIM(k)$ relating the efficiency of the managed system at two different impact states is then defined as:

$$MIM(k_0, k) = 1 - \frac{E(k)}{E(k_0)} \in [0, 1] \tag{5}$$

This is the impact function that is used through the paper. The intuition behind our function is to capture the behavior of the performance of the managed system. This behavior is observed from a baseline configuration and define which cases the performance of the system is unacceptable under a management impact factor. The efficiency $E(k_0)$ denote a baseline configuration of the managed system with a value k_0 of the impact factor. A way to determine the baseline configuration is to suspend all management activities for a period and measure the performance of the target managed system during that period as the baseline. In this case, the value of the manageability $G(0) = 0$ and $E(0) = 1$. Then, for the baseline configuration of the managed system, the baseline efficiency is equal to 1 and the impact function is given by :

$$MIM(k) = 1 - E(k) = 1 - \frac{F(k)}{F(k) + G(k)}; \text{ where } k \geq 1 \tag{6}$$

From the management efficiency aims, a managed system is isoefficiency managed if its overall efficiency is maintained at a desired value such as $0 \leq E(k) \leq 1$ which implies that the useful work performed by the managed system (productivity) should grow at least at the same rate as the management overhead (manageability) to keep managed system efficiency constant. Let α denote the value of $G(k)$ normalized with respect to $F(k)$. α denotes the fraction from the managed system productivity attributed to the management activities. Then, $G(k) = \alpha.F(k)$ and we obtain $MIM(k) = \frac{\alpha}{1+\alpha}$.

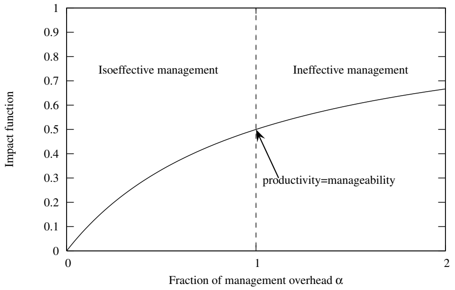


Fig. 2. The impact metric behavior of a linear model between manageability and productivity

Figure 2 depicts the behavior of the impact metric according to a linear fashion of manageability work and the managed system productivity. When $\alpha = 1$, the manageability has the same rate as productivity and in this case we reach the bound of the isoeffective management. Beyond that value, the management strategy becomes ineffective.

4.1 Computing the Impact Function

Calculation of this metric depends on the performance technique used to evaluate both the performance of the managed system and the management one. Analytical and simulation techniques are more suitable to calculate it. These two techniques are more flexible [13] than the measurement technique and they can handle a wide variety of configurations of the managed system by varying the impact factor and management profiles. Their disadvantage is that they need the availability of analytical models both for the managed system and the management one. It is not easy to obtain them for complex distributed systems. We define the following steps to calculate the impact function. We first determine the baseline performance of the target system. The baseline value will capture the performance of the system under fixed states of user's work and scalability values (e.g, a fixed number of users, a fixed number of requests per unit of time, a fixed number of servers, etc). By varying the baseline configuration of the managed system we can capture its performance under different users workload or scalability factors. Secondly, we define the management profile of the managed distributed system and the impact factor. The productivity of the managed system and the manageability of the management one are computed as follows:

1. Fix a management profile which includes management strategy parameters as described in section 3.
2. Choose an impact factor and fix other factors.
3. Managed system productivity prediction: the productivity is computed by measuring the average system throughput in number of responses per second and the average response time per response. Measure the running cost per second on the managed system. The cost is the sum of resources utilization expressed on a measurement unit (e.g., average percent).
4. Management system manageability prediction : needs measurement of the average management throughput in number of management responses per second from the

agent and the average response time. The cost represents the overhead running cost per second due to management activities.

5. Compute the impact function according to equation 2 in case of use of the value function $f(k)$ or equation 4 in case of use of only the response time $T(k)$.
6. Vary the chosen impact factor value and goto the step 3.

Computing the impact metric by varying the impact factor k within an interval allows us to find its bound value, beyond which the management highly impacts the productivity. In that case, a stronger justification for its benefit on the overall service delivery is required. The bound value of the impact factor k correspond to an impact metric value equal to 0.5.

5 Experimental Assessment

To assess the applicability of our impact metric, we did study it in the context of a JMX based management of a J2EE application server such as JBoss [14]. The management profile that we evaluate from the MBean server within the JBoss server is the following management profile

- Client-Server approach,
- JMX based management,
- Monitoring task,
- Polling based monitoring by using *getAttribute* operation.
- The JMX agent is deployed as a driver: the MBean server within the JBoss server is implemented as the kernel of the server,
- The management instrumentation is internal and the attribute that we solicit is retrieved from the system. The *getFreeMemory* operation, at the JMX level, calls the JVM system function *Runtime.freeMemory* to retrieve the amount of free memory in the JVM.

We take the management input workload expressed in number of requests per second as the impact factor. Previous work [15] gives us an idea of a realistic range for this factor. We vary the management workload from 1 to 400 requests/s. If we go beyond this value, the number of lost management requests increase consequently due to timed-out RMI connections (we keep the default value of 15 seconds of the RMI timeout).

5.1 Testbed Setup

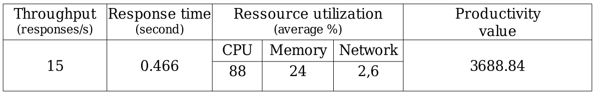
We used a JBoss v3.2.1, running on a Sun JDK v1.4.2 and hosted on a bi-processor PIII 550MHZ with 512MB RAM, with the Slackware 9.1 operating system. The testbed workstations are connected to a 100Mbps Ethernet switch. The testbed is alone on this network, and we can assume that network is not a limitation. To emulate users activity against the JBoss server, we use RUBiS [16] as a benchmarking tool to evaluate the performance of the JBoss server and to measure its productivity. RUBis is modeled after an auction site (eBay.com). For our experiments, we chose to use an EJB variant from RUBiS, which is entirely based on stateless session beans, as it is the best performing

EJB variant according to [17]. For the measurements shown here, we used a steady users workload. The number of emulated users is kept constant (100 clients) and they have a mean *thinking time* of 7 seconds. To capture the management impact, we developed an emulator management client that implements only the monitoring task. The management emulator client sends a number of requests per second, that represents the impact factor, by using the *getAttribute* operation exposed by the MBeanServer within the JBoss server. We retrieve the value of the *FreeMemory* attribute from the *ServerInfo* MBean. In the current work, all management requests solicited the same MBean and the same attribute. Each measurement has a duration of 15 minutes and a warm-up period of 2 minutes both for users emulator client and management emulator client. Our experiments proceed as follows. The set of 100 emulated users were running. They execute browsing-only transactions against the JBoss server. The response time and the average number of responses per second is measured on the users client emulator side. Concurrently to users workload, the management client emulator executes a fixed number of *getAttribute* requests per second against the MBeanServer within the JBoss server. On the management client side, we measure the number of management responses per second and the response time per request. We use the *sar* [18] tool to measure the resource utilization (cpu,memory and network) on the JBoss server side. Response times measurements are taken using the *System.currentTimeMillis()* method included in the API of Sun's JDK.

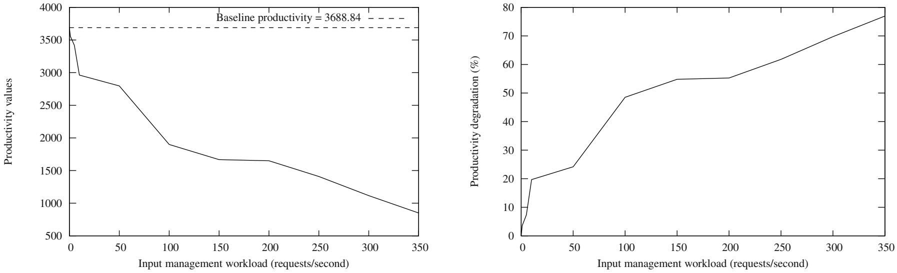
5.2 Experimental Results

In a first step, we measure the baseline performance of the JBoss server without management workload and where only the users steady workload is executed against the server. Table 2 displays baseline average values of the throughput, response time and resource utilization and their corresponding baseline productivity of the JBoss server. In a second step, we vary the management input workload and measure the productivity, the manageability and the impact metric at each impact factor value. From the plot of figure 3 we can observe a decrease of the server productivity values due to the high management input load and the management overhead within the server. The productivity degradation from the baseline configuration, where only the users workload is executed against the JBoss server, varies between 24% for 50 management requests per second and 74% for 400 requests/s. The JBoss server productivity degradation is caused by the increase of response times $T(k)$ and the decrease of the throughput $\lambda(k)$ as shown in figure 4. From figure 5, we see the increase of manageability. This is trivial, due to the increase of the management load. Figure 6 shows the increase of the impact metric.

Table 2. Results of the baseline productivity of the JBoss server without any ongoing monitoring activity and under a steady users workload (100 browsing clients)

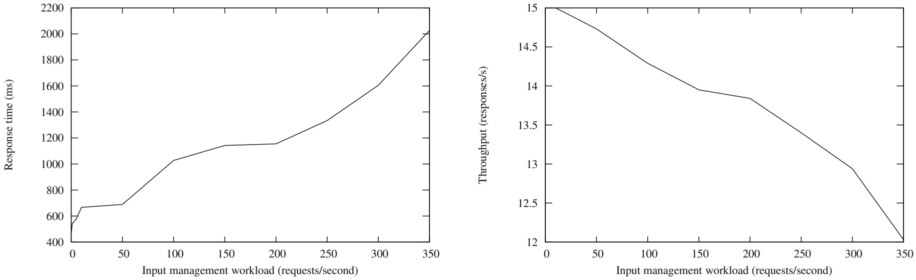


Throughput (responses/s)	Response time (second)	Ressource utilization (average %)			Productivity value
		CPU	Memory	Network	
15	0.466	88	24	2,6	3688.84



(a) JBoss server productivity decrease **(b) JBoss server productivity degradation in (%)**

Fig. 3. Productivity decrease of the JBoss server (a) and the corresponding degradation from the baseline server state for a steady users workload and under an increasing management workload input in number of requests per second using the *getAttribute* operation that retrieves the FreeMemory attribute from the ServerInfo MBean



(a) JBoss server response times increase **(b) JBoss server throughput decrease**

Fig. 4. Response times increase (a) and throughput decrease (b) of the JBoss server under an increasing management workload and a steady users workload (100 browsing clients)

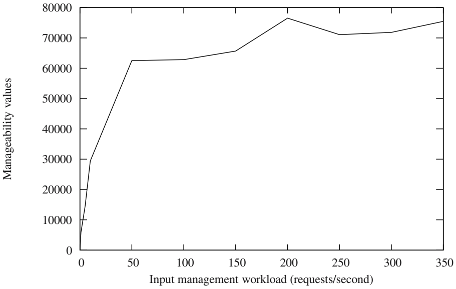
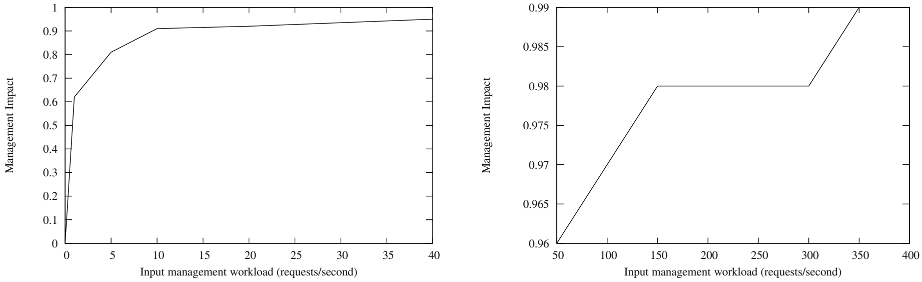


Fig. 5. Manageability growth by increasing the management workload input in requests per second by using the *getAttribute* operation that retrieves the FreeMemory attribute from the ServerInfo MBean



(a) Low management workload **(b) High management workload**

Fig. 6. Management Impact Metric behavior for the JBoss server under a low management workload input less than 50 req/s (a) and a high management workload input greater than 50 req/s (b) by using the *getAttribute* operation that retrieves the FreeMemory attribute from the ServerInfo MBean

Here we note that the management agent (the MBeanServer from the JMX terminology) of the JBoss server is the core of the application server (driver model). It is the components container for the server design level (not ejb level). Thus, the input management workload is executed concurrently with user's workload and the impact is quickly seen.

6 Conclusion and Future Work

We defined a metric that captures the impact of a management profile or strategy on the performance of a managed system. The objective is to provide a metric that evaluates a management strategy and enables it to support its intended target environment. Our experiments confirmed that a management strategy within a managed system is associated with a degradation of the overall system efficiency, which may not be acceptable in all cases. Here we should note that our experiments overestimated the manageability, because we take into account a high management input rates and we use the same value of the running resource consumption both for computing manageability and productivity on the server side, which explains the fast growth of the impact metric. A more accurate analysis and estimation of the running resource utilization of manageability should be done by using complementary techniques such as the profiling component resource consumption presented in [19]. Our impact metric is computed using the measurement technique to evaluate the performance of the managed JBoss server. This technique needs a lot of time to collect measurements to be credible and requires available system prototypes. Other techniques (analytical modeling or simulation) are more flexible and less time consuming than the measurement technique [13]. They will be used for evaluating the performance of the managed system under a given management impact factor and computing the impact metric. In this work, we examine only the case of a degradation impact of management activities on the performance of a managed system. We plan to investigate other cases where the impact of management activities, such as load balancing and admission control, might improve and maintain the performance of a managed system. In parallel, we continue to setup and run performance tests of the mannagemet plane on large sale systems e.g. Grids to massively deploy agents and evaluate the behavior of both manager and agent systems.

References

1. Pavlou, G., Flegkas, P., Gouveris, S., Liotta, A.: On management technologies and the potential of web services. Communications Magazine, IEEE **42** (2004) 58–66 ISSN: 0163-6804.
2. Neisse, R., Vianna, R.L., Granville, L.Z., Almeida, M.J.B., Tarouco, L.M.R.: Implementation and bandwidth consumption evaluation of SNMP to web services gateways. In: NOMS (Network Operations & Managament Symposium). Volume 9. (2004)
3. Jogalekar, P., Woodside, C.: Evaluating the scalability of distributed systems. IEEE Trans. Parallel Distrib. Syst. **11** (2000) 589–603
4. Burness, A., Titmuss, R., Lebre, C., Brown, K., Brookland, A.: Scalability evaluation of a distributed agent system. Distributed Systems Engineering **6** (1999) 129–134
5. Pattinson, C.: A study of the behaviour of the simple network management protocol. In: 12th International Workshop on Distributed Systems: Operations and Management (DSOM). (2001) 305–314
6. Subramanyan, R., Miguel-Alonso, J., Fortes, J.: A scalable SNMP-based distibuted monitoring system for heterogeneous network computing. In: Proceedings of the 2000 ACM/IEEE conference on Supercomputing, IEEE Computer Society (2000) 14
7. 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)
8. Mitra, A., Maheswaran, M.: Measuring scalability of resource management systems. Technical Report SOCS-04.5, School of Computer Science,McGill University (2004)
9. Diao, Y., Hellerstein, J., Parekh, S., Griffith, R., Kaiser, G., Phung, D.: Self-managing systems: A control theory foundation. In: 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), Greenbelt, Maryland (2005) 441–448
10. Murray, J.: Designing manageable applications. WEB Developper's Journal (2003)
11. Kreger, H., Harold, W., Willamson, L.: Java and JMX: Building Manageable Systems. Addison-Wesley (2003) ISBN: 0672324083.
12. Grama, A., Gupta, A., Kumar, V.: Isoefficiency function: A scalability metric for parallel algorithms and architectures. IEEE PDT **1** (1993) 12–21
13. Jain, R.: The art of Computer Systems Performance Analysis. John Wiley & Sons, Inc (1991) ISBN : 0-471-50336-3.
14. JBoss: The professional open source company. http://www.jboss.org (1999)
15. Lahmadi, A., Andrey, L., Festor, O.: Performances et résistance au facteur d'échelle d'un agent de supervision basé sur jmx : Méthodologie et premiers résultats. In: Colloque GRES 2005 : Gestion de REseaux et de Services, Luchon, France. Volume 6. (2005) 269–282 ISBN : 2-9520326-5-3.
16. ObjectWeb: Rubis: Rice university bidding system. http://rubis.objectweb.org (2002)
17. Cecchet, E., Marguerite, J., Zwaenepoel, W.: Performance and scalability of ejb applications. In: Oopsla'02. (2002) http://rubis.objectweb.org/download/perf_scalability_ejb.pdf.
18. Godart, S.: system performance tools for linux os. http://perso.wanadoo.fr/sebastien.godard/ (2003)
19. Stewart, C., Shen, K.: Performance modeling and system management for multi-component online services. In: The 2nd Symposium on Networked System Design and Implementation (NSDI2005),Boston, MA, USENIX (2005)

Improving the Configuration Management of Large Network Security Systems

João Porto de Albuquerque[1,2,⋆], Holger Isenberg[2], Heiko Krumm[2], and Paulo Lício de Geus[1]

[1] Institute of Computing, State University of Campinas, 13083-970, Campinas/SP Brazil
{jporto, paulo}@ic.unicamp.br

[2] FB Informatik, University of Dortmund, 44221 Dortmund Germany
{joao.porto, heiko.krumm, holger.isenberg}@udo.edu

Abstract. The security mechanisms employed in today's networked environments are increasingly complex and their configuration management has an important role for the protection of these environments. Especially in large scale networks, security administrators are faced with the challenge of designing, deploying, maintaining, and monitoring a huge number of mechanisms, most of which have complicated and heterogeneous configuration syntaxes. This work offers an approach for improving the configuration management of network security systems in large-scale environments. We present a configuration process supported by a modelling technique that uniformly handles different mechanisms and by a graphical editor for the system design. The editor incorporates focus and context concepts for improving model visualisation and navigation.

1 Introduction

In today's large networked environments security is a major concern. A great variety of security technologies and mechanisms are employed in these environments in order to offer protection against network-based attacks. Whilst significant progress has been made on improving network security technology in recent years, only quite modest attention has been given to its configuration interface. In practice, a security administrator must deal with a variety of complex and heterogeneous configuration syntaxes, most of which are unintuitive and in some cases even misleading.

The situation is especially dramatic in large-scale environments where it is very hard to have a global view of the numerous security mechanisms that have to be put into harmonic cooperation. In these situations, a single maladjustment between two mechanisms can leave the whole system vulnerable. Approaches that offer proper abstraction, integration and tool support for managing the configuration of security mechanisms are thus key factors for making the configuration process less error-prone and more effective.

⋆ Scholarship funding by the German Academic Exchange Service (DAAD).

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 36–47, 2005.

Four basic tasks of the network security configuration management can be distinguished: i) the design of the security system, including the definition of technologies and mechanisms to be employed, as well as the placement of the diverse security components over the network; ii) the deployment of the designed configuration; ii) configuration maintenance, enabling the introduction of changes to achieve adaptability in face of new requirements; iv) monitoring of the system during run-time to assure compliance with expected behaviour.

This paper addresses the three first phases of the configuration management process. To support the design phase, we use a modelling technique that allows the design of the security system to be managed in a modular fashion, by means of an object-oriented system model [8]. This model is segmented into logical units (so-called *Abstract Subsystems*) that enclose a group of security mechanisms and other relevant system entities, and also offer a more abstract representation of them. In this manner, the system administrator is able to design a security system—including its different mechanism types and their mutual relations—by means of an abstract and uniform modelling technique.

A software tool supports the modelling, providing a graphical editor. This editor incorporates the concept of *focus & context*—that originated from research on information visualisation—through the techniques of fisheye-view [11] and semantic zooming [3,7]. Furthermore, our work builds upon the policy hierarchy [6] and model-based management [5] approaches in order to assist the above-mentioned configuration management phases of deployment and maintenance. A system model organised in different abstraction layers thus affords a step-wise, tool-assisted system modelling, along with an automated policy refinement that culminates in the generation of low-level configuration parameters. While this guided derivation of parameters contemplates the deployment of the security configuration, its maintenance is supported by the possibility of editing the models and repeating the automated generation process.

The rest of the paper is organised as follows: Sect. 2 presents the main elements of our modelling technique, and Sect. 3 describes the *focus & context* techniques incorporated into the support tool. Subsequently, Sect. 4 presents a case study that exemplifies the practicality of these approaches within a typical large-scale networked environment. In Sect. 5 our results are compared to related work. Finally, Sect. 6 casts conclusions for this paper.

2 Modelling Technique

Our modelling builds upon the Model-based Management approach [5] and employs a three-layered model whose structure is shown in Fig. 1. The horizontal dashed lines of the figure delimit the abstraction levels of the model: *Roles & Objects* (RO), *Subjects & Resources* (SR), and *Diagram of Abstract Subsystems* (DAS). Each of these levels is a refinement of the superior one in the sense of a "policy hierarchy" [6]; i.e. as we go down from one layer to another, the abstract system's view contained in the upper level is complemented by the lower-level system representation, which is more detailed and closer to the real system. As

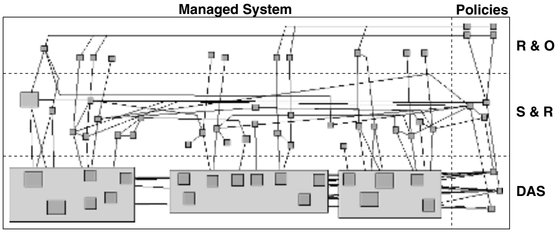


Fig. 1. Model Overview

for the vertical subdivision, it differentiates between the model of the actual managed system and the security policies that regulate this system.

As the lowest level of the model (DAS) is the focus of the present work, it is explained in detail in the next section. The two uppermost levels (RO and SR) have been adopted from previous work on model-based management, and thus will be presented briefly.

The RO level is based on concepts from Role-Based Access Control (RBAC) [10]. The main classes in this level are: *Roles* in which people who are working in the modeled environment act; *Objects* of the modeled environment that should be subject to access control; and *AccessModes*; i.e. the ways of accessing objects. The class *AccessPermission* expresses a security policy, allowing the performer of a *Role* to access a particular *Object* in the way defined by *AccessMode.*

The second level (SR in Fig. 1) offers a system view defined on the basis of the services that will be provided, and it thus consists of a more complex set of classes. Objects of these classes represent: (a) people working in the modeled environment (*User*); (b) subjects acting on the user's behalf (*SubjectTypes*); (c) services in the network that are used to access resources (*Services*); (d) the dependency of a service on other services (*ServiceDependency*); and lastly (e) *Resources* in the network.

2.1 Diagram of Abstract Subsystems

The main objective of the Diagram of Abstract Subsystems (DAS) is to describe the *overall structure* of the system in a modular fashion; i.e. to cast the system into its building blocks and to indicate their interconnections. As such, a DAS is, formally speaking, a graph comprised of Abstract Subsystems (ASs) as nodes and edges that represent the possibility of bi-directional communication between two ASs. An AS, in turn, contains an abstract view of a certain system segment; i.e. a simplified representation of a given group of system components that may rely on the following types of elements:

Actors: groups of individuals which have an *active* behaviour in a system; i.e. they initiate communication and execute mandatory operations according to *obligation policies.*

Mediators: elements that intermediate communication, in that they receive requests, inspect traffic, filter and/or transform the data flow according to the *authorisation policies*; they can also perform mandatory operations based on *obligation policies*, such as registering information about data flows.

Targets: *passive* elements; they contain relevant information, which is accessed by *actors*.

Connectors: represent the interfaces of one AS with another; i.e. they allow information to flow from, and to, an AS.

Each element of the types *Actors*, *Mediators* or *Targets* represents a group of system elements that have a relevant behaviour for a global, policy-oriented view of the system. As for the *Connectors*, they are related to the physical interfaces of an AS (for a detailed explanation on the modelling of abstract subsystems we refer to [8]). In this manner, a DAS supports the reasoning about the structure of the system *vis-à-vis* the executable security policies, thus making explicit the distribution of the different participants of these policies over the system.

Furthermore, in order to model the security policies themselves, another object type is also present in a DAS: *ATPathPermissions* (ATPP). An ATPP is associated with a path (p) in a DAS that connects an *Actor* (A) to a *Target* (T), possibly containing a number of *Mediators* and *Connectors* along the way. It expresses the permission for p to be used by A in order to access T. In this manner, each ATPP models an *authorisation policy*. The ATPP objects in the system are not defined by the modeller, but rather derived automatically in a process that is explained later in Sect. 4.4.

Additionally, each AS in a DAS is also associated with a detailed view of the system's actual mechanisms. This expanded view encompasses objects that represent hosts, processes, protocols and network interfaces of the system (this detailed view is related to the level PH of the works on model-based management [5]) and supports the process of automated generation of configuration parameters (Sect. 4.4).

Example of DAS. An example of DAS is shown in Fig. 2. This diagram corresponds to a simple network environment with three ASs: "internal network",

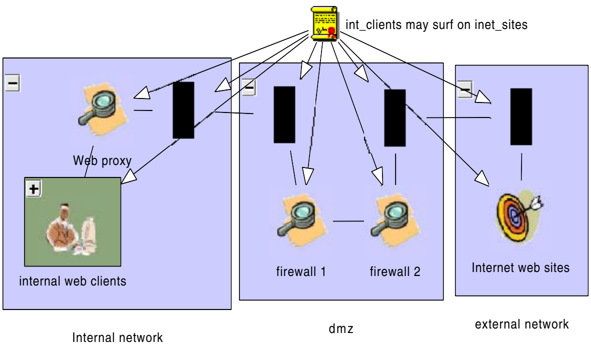


Fig. 2. Example of DAS

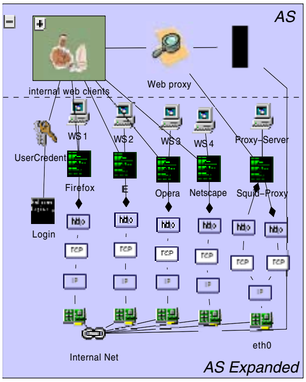


Fig. 3. Expanded AS

"dmz" (*demilitarised zone*) and "external network". In the "internal network", the object "internal web clients" is an *Actor* representing a group of processes that are authorised to access the processes mapped by the *Target* "internet web sites" (in the "external network") through the *Mediator* "Web proxy", by the ATPP "int_clients may surf on inet_sites".

Figure 3 shows both the abstract and the expanded view for the AS "internal network" (leftmost of Fig. 2). Each object in the abstract view of the AS is then related to the elements that model the corresponding real entities of the system; for instance, the *Actor* "internal web clients" is associated with its respective *Process*-typed objects. This double representation of an AS provides the designer with a flexible model of the system that offers not only a more abstract, concise and understandable description of the system's structure, but also a detailed view of its mechanisms. It also constitutes the basis for applying the techniques presented in the next section.

3 Focus and Context

The term *focus & context* refers to techniques that allow a user to centre his view on a part of the model that is displayed in full detail (*focus*), while at the same time perceiving the wider model surroundings in a less detailed manner (*context*). The major advantage of using these techniques is the improved space-time efficiency for the user; i.e. the information displayed per unit screen area is more useful and, consequently, the time required to find an item of interest is reduced as it is more likely to be already displayed [7]. We employ the *focus & context* concept within two different methods, described in the next sections in turn.

3.1 Semantic Zooming

The concept of *semantic zooming* [7,3] is based on the ability to display model objects in different abstraction levels, depending on their distance from the focus. Thus, objects inside the focused region of a diagram are exhibited in their full detailed form, whereas objects located at the borders are shown in the most simplified way. The regions between these two extremes are displayed with intermediary levels of detail. In this manner, the presented information is selectively reduced by adjusting the level of detail in each region to the user's interest in this region.

In our context, there are two classes of compounded objects to which semantic zooming is applied: *typed folders* and *Abstract Subsystems.* A *typed folder* is an object that aggregates a group of objects of same class (or type), for the sake of the representation conciseness. On the other hand, *Abstract Subsystems* contain objects of various classes (Sect. 2.1) and may also enclose *typed folders.* In BOTH cases, the level of detail shown can be changed by the selective display of internal objects.

In the simplest situation, two different representations of typed folders are available: a closed (all internal objects are hidden) and an open folder view. As for the ASs, three different levels of abstraction are used: i) a full detailed view that includes all the internal objects (as in Fig. 3); ii) an abstract representation encompassing only the objects of the abstract view (as in Fig. 2); and iii) a "closed" view in which none of the internal objects are displayed.

3.2 Fisheye View

The term *fisheye view* is used for the type of projection created by a fisheye lens used in photography. This type of lenses achieves a 180° field of view and is uncorrected. It results in an optical enlargement of objects near the centre in relation to those at the borders. This feature emulates the human visual perception, which by the effect of the eye movements has a clear focused area and a gradual loss of visual resolution in the direction of the peripheral regions. A fisheye view combines thereby a complete image overview with a gradual degradation of detail that increases with the distance from the focus—and it is thus well suited to implement the concept of *focus & context*. In contrast to semantic zooming, the fisheye view manipulates the displayed size of the objects in order to change the amount of information shown.

We build upon the practical application of fisheye views for graph visualisation offered by Sarkar and Brown [11]. In our tool, the focus area can also be freely moved by the user throughout the model. In this way, objects within the focused area are displayed in an enlarged scale whereas the others become gradually smaller as they are approach the model borders.

4 Configuration Design and Deployment Process

To illustrate the practical application of the previously described concepts, in this section we analyse a paradigmatic case study. The considered scenario is that of an enterprise network, composed of a main office and branch office, that is connected to the Internet. Our main goal is to assist the security administrator in the task of designing the configuration for the security mechanisms that are required to enable and control web-surfing and e-mail facilities for the company's office employees.

Therefore, the highest-level security policies for this environment can be stated as follows: **P1**: The employees may surf on the Internet from the computers in the main and branch offices; **P2**: The employees may read their internal e-mails from the main and branch offices, and from home; **P3**: The employees may send e-mail to external and internal addresses; **P4**: Internet users may access the corporate web server; **P5**: Internet users may send e-mail to internal mail addresses.

Having as input the above abstract policy statements and previously described scenario, we apply our modelling technique by providing in the next sections a step-by-step description of the configuration design process, passing through the different abstraction layers.

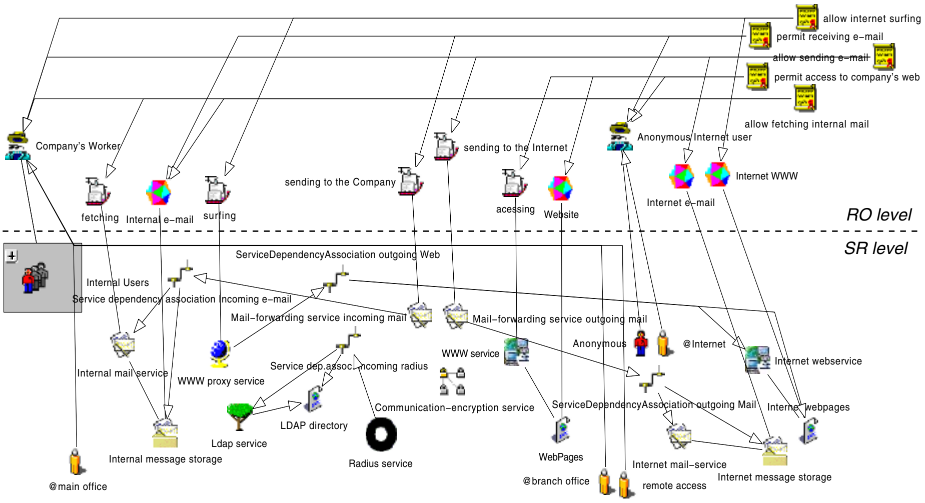


Fig. 4. Model of the levels RO and SR

4.1 Modelling of the RO Level

Since the highest level in our model is based on RBAC concepts (Sect. 2), the designer starts the development process by mapping the abstract policies, expressed in natural language, to the more formal syntax of RBAC. The top of Fig. 4 shows the resulting model at the RO level for our considered scenario. The basic objects are: the *Roles* "Company's Worker" and "Anonymous Internet User", and the *Objects* "Internal e-mail", "Website", "Internet e-mail" and "Internet WWW". These objects are associated to *AccesModes* by means of five *AccessPermissions* (at the top, on the right of Fig. 4), each of the latter corresponding to one of the abstract policy statements of the previous section. Thus, for instance, the *AccessPermission* "allow Internet surfing" models the policy statement **P1**, associating the role "Company's Worker" to "surfing" and "Internet WWW". The other policy statements are analogously modeled by the remaining *AccessPermissions*.

4.2 Modelling of Users, Services and Resources

The second step in the design process consists of the definition of the services that the system must provide, the resources they need, and the users who may take advantage of them. In our example, the *User* "Anonymous" and the *SubjectType* "@Internet" are defined in association with the role "Anonymous Internet User". For the role "Company's Worker", several *User* objects are grouped in the *TypedFolder* (Sect. 3.1) "Internal Users", and three *SubjectTypes* are defined: "@main office", "@branch office" and "@remote access" (at the bottom of Fig. 4). These objects map the three types of session that can be established by an employee in the considered scenario, depending on his physical location.

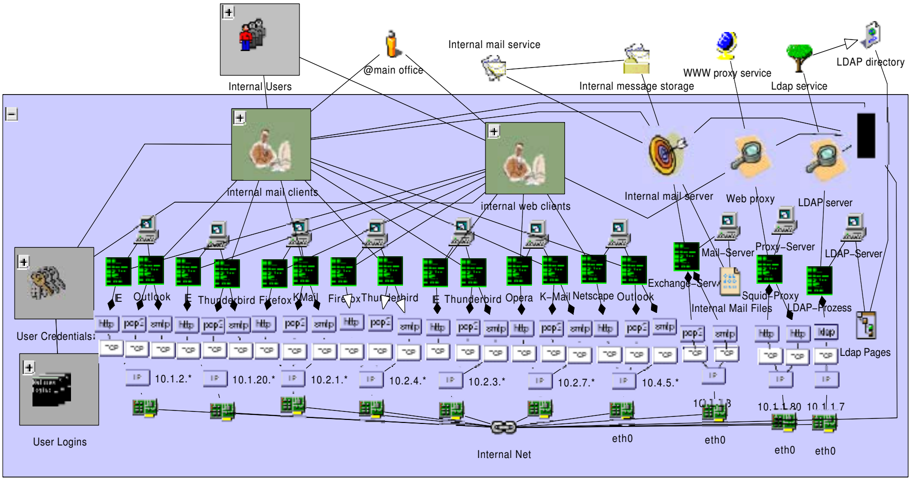


Fig. 5. Extract of a DAS and its relation to the SR level

As for the modelling of services and resources, a *Service* object will be basically defined for each *AccessMode* in the RO level, whereas RO *Objects* will be mapped to SR *Resources*. In case where more than one service is needed to provide access to a resource, this fact is expressed by a *ServiceDependency*. This is what happens in our example model, for instance, with the *AccessMode* "sending to the company" that is related to the *Service* "Mail-forwarding service incoming mail". This service must rely on the "Internal mail service" in order to provide access to the resource "Internal message store" (which is associated with the RO *Object* "Internal e-mail").

4.3 Modelling of Abstract Subsystems

In order to produce a Diagram of Abstract Subsystems, the designer shall start with the identification of the major segments in which the system is subdivided. Considering our example and also accounting consolidated network security techniques, a structural subdivision into five blocks can be defined: the internal network, the demilitarised zone (dmz), the branch office network, the remote access points and the remaining external network (the Internet). Therefore, the DAS for this example has an AS for each one of these segments.

Afterwards, the modeller must define, for each subsystem, the security mechanisms and other relevant network elements with respect to the security policies; i.e. to model the expanded view of each AS (Sect. 2.1). The bottom of Fig. 5 shows the expanded AS "internal network". It has objects that represent processes running on seven workstations, one mail server, one web proxy and one LDAP server. Each one of these processes is connected through the adequate protocol stack—modeled by a series of interconnected corresponding protocol objects—to their network interfaces.

Subsequently, the abstract view of each AS must be defined by the creation of objects for *Actors*, *Mediators*, *Targets* and *Connectors*, and their associations to the objects of the SR level must be established. This is accomplished by classifying the behaviour of the elements of the expanded view into one of these classes (the mapping of the abstract view in ASs is further elaborated in [8]).

The *Actors* "internal mail clients" and "internal web clients" (see Fig. 5) are created in the "internal network" to map the processes of this subsystem with active behaviour in our example. They are also both connected to the objects in the SR level that map the same behaviour: "Internal Users" and "@main office". Due to their intermediary or supporting functions, the *Mediators* "Web proxy" and "LDAP Server" are created and connected to the corresponding processes in the expanded view; they also connect these processes with the appropriate services in the SR level. Proceeding in an analogous manner for all of the remaining ASs in our example, a complete DAS is achieved and the whole system model is complete.

4.4 Policy Refinement and Configuration Generation

After inputting all model levels, the system design phase is complete and the security administrator can take advantage of our support tool to deploy the configuration parameters for the security mechanisms modeled. This is accomplished by an automated building of a policy hierarchy, starting from the abstract security policy defined by the designer in the RO level (Sect. 4.1) and deriving lower-level policies on the basis of the model entities of the levels SR and DAS.

Thus, the support tool first derives each one of the given *AccessPermissions* through a series of intermediary objects, ultimately generating a set of *ATPathPermissions* (Sect. 2.1). Finally, for the last step of the configuration deployment, a series of back-end modules are executed, where each module corresponds to a special security service product (e.g. Kerberos, FreeS/WAN, Linux IP tables etc.). These back-end functions evaluate the *ATPathPermissions* and the expanded views of the ASs in order to automatically generate the adequate configuration files for each of the security mechanisms (an extensive explanation on the policy refinement process is beyond the scope of this paper and can be found in [9,5]).

4.5 Model Editing, Navigation and Visualisation

During the model editing needed to accomplish the tasks described in the previous sections, in order to edit specific parts of large models (such as the one used for our case study), a designer must rely on model cutout enlargement techniques; i.e. on zooming. However, with the standard method of zooming that is based on linear enlargement of a fixed-size model cutout, the model navigation and visualisation are problematic. Consider, for instance, the simple design task of connecting an object in the model area that is currently edited to another one that is located at the opposite extreme of a large model. In this case, "large" means that the whole model does not fit into the screen when scaled to a size that makes its editing possible. With the standard zooming method one needs to perform the following steps: 1) scale down, in order to be able to see the entire model; 2) estimate the locations of the target and source objects in the out-zoomed view, and the angle of the edge needed to connect them; 3) enlarge the area around the source object, in order to be able to select it; 4) select the source; 5) drag a new edge from the source into the direction of the previously estimated angle (the enlarged region of the model moves automatically following the mouse); 6) stop the dragging once the target can be seen on the screen; 7) drop the edge on the target object.

On the other hand, with the use of a fisheye view only the following steps are needed: 1) activate the fisheye view mode; 2) select the source object, which is displayed inside the enlarged focus area (as in the left hand picture of Fig. 6); 3) drag a new edge from the source into the direction of the target, whose location can be simultaneously seen in the down-scaled surroundings; (the focus follows the mouse during this process, so that the target can eventually be seen in detail, as in the right hand picture of Fig. 6); 4) drop the edge on the target object.

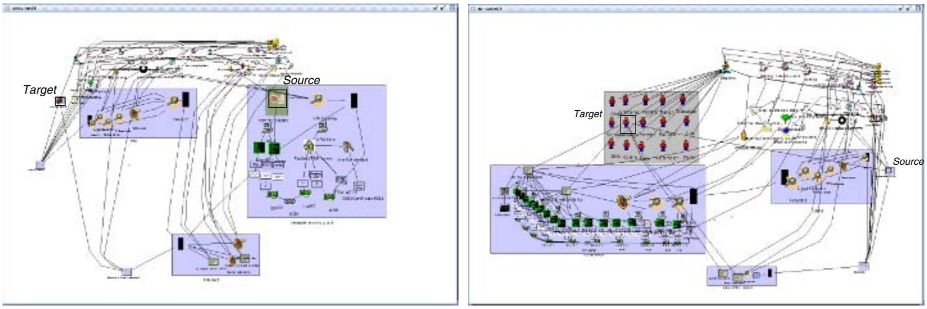


Fig. 6. Fisheye view with focus centred at the source and target nodes

Therefore, using a fisheye view reduces the number of steps needed by almost half. Furthermore, the usage is immediately intuitive, since the designer never loses of sight the full context of the model, and the non-linear adaptive scaling seems “natural” in contrast to the sharp border between linear enlarged cutouts and their surroundings.

Combined Focus and Context. Since the two techniques introduced in Sect. 3 operate on orthogonal subjects—namely, fisheye view on graphics and semantic zooming on structure—they can be combined. We accomplish this by using the scale factor resulting from the fisheye transformation function to adjust the abstraction level which is used to display compound elements.

The result of this is that as the distance between the focus and a certain object increases, this object is gradually presented in a more abstract view—which *per se* has a smaller graphical representation—and also graphically miniaturised by the fisheye function. Thus, a larger focused area is made possible even if the context is still visible, which leads to an optimisation of the screen space.

In Fig. 6, the effect of this combined use can be seen. In the left hand picture, the AS "remote access point" (that encloses the source node) has the focus: it is thus displayed in a higher scale and in its full level of details, allowing editing. The ASs "dmz" and "Internet" pertain to a close context and are shown in their abstract representation, gradually smaller in size; whereas the miniaturised and "closed" views of the remaining ASs save screen space while still enabling the user to perceive their existence.

5 Related Work

Though there are several applications of *focus & context* techniques for improving the usability of generic graph editors (including the recent applications to UML in [7] and the more generic approach in [3]), as far as we know, they have not yet been used in the context of model visualisation and navigation for network security system design.

In a wider context, Damianou *et al.* [2] present a set of tools for the specification, deployment and management of policies specified in the Ponder language. The tool prototype includes a domain browser that uses fisheye views to handle large structures. While centring the approach in the policies, this work does not provide a representation of the architecture of the system to be managed, making it hard for the system designer to associate the policies with his/her mental model of the system. A further work by these authors offers an approach to the implementation and validation of Ponder policies for Differentiated Services using CIM to model network elements [4]. CIM concentrates on the modelling of management information (e.g. device's capabilities and state) while our model represents the whole relevant structure of the managed system together with the management components.

The graphical tool Firmato [1] seems to be the closest approach to ours, since it supports the interactive policy design by means of diagrams and automatically derives the corresponding configurations for mechanisms. However, since the abstraction levels of policy definitions and configuration parameters are relatively near to each other, its support is restricted to an abstraction level that is close to the mechanisms.

6 Conclusion

This paper has presented an approach for the configuration management of large network security systems that builds upon and extends previous work on Model-based Management [5] and on the Diagram of Abstract Subsystems [8,9]. We add to these works by proposing a general design process that is centred around the system administrator and encompasses four steps that range from the abstract policy modelling to the generation of low-level configurations. In this manner, the abstract policy representation is gradually brought into a more concrete system view, bridging the gap between high-level security policies and real system implementation.

We also combine the use of a modelling technique specially conceived to achieve scalability with *focus and context* techniques, thereby allowing the designer to define in detail a certain model part without losing sight of the system as a whole. Therefore, we expect our methodology to contribute to making the configuration management of security technologies in large-scale network environments more effective and closer to the security administrator. Current work concentrates on the representation of policies at the lower model levels, in order to enhance their handling.

References

1. Y. Bartal, A. J. Mayer, K. Nissim, and A. Wool. Firmato: A novel firewall management toolkit. *ACM Transactions on Computer Systems*, 22(4), November 2004.
2. N. Damianou, N. Dulay, E. Lupu, M. Sloman, and T. Tonouchi. Tools for domain-based policy management of distributed systems. In *IEEE/IFIP Network Operations and Management Symposium (NOMS2002)*, Florence, Italy, 2002.
3. O. Köth and M. Minas. Structure, abstraction, and direct manipulation in diagram editors. In *Diagrammatic Representation and Inference, Second International Conference (Diagrams 2002)*, volume 2317 of *Lecture Notes in Computer Science*, Callaway Gardens, GA, USA, 2002. Springer.
4. L. Lymberopoulos, E. Lupu, and M. Sloman. Ponder policy implementation and validation in a CIM and differentiated services framework. In *IFIP/IEEE Network Operations and Management Symposium (NOMS 2004)*, Seoul, Korea, April 2004.
5. I. Lück, S. Vögel, and H. Krumm. Model-based configuration of VPNs. In *Proc. 8th IEEE/IFIP Network Operations and Management Symposium NOMS 2002*, pages 589–602, Florence, Italy, 2002. IEEE.
6. J. D. Moffett and M. S. Sloman. Policy hierarchies for distributed system management. *IEEE JSAC Special Issue on Network Management*, 11(9), 11 1993.
7. B. Musial and T. Jacobs. Application of focus + context to UML. In *Australian Symposium on Information Visualisation*, volume 24 of *Conferences in Research and Practice in Information Technology*, Adelaide, Australia, 2003. ACS.
8. J. Porto de Albuquerque, H. Krumm, and P. L. de Geus. On scalability and modularisation in the modelling of security systems. In *10th European Symposium on Research in Computer Security (ESORICS 05)*, volume 3679 of *LNCS*, pages 287–304, Heidelberg, Germany, September 2005. Springer Verlag.
9. J. Porto de Albuquerque, H. Krumm, and P. L. de Geus. Policy modeling and refinement for network security systems. In *POLICY '05: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)*, pages 24–33, Washington, DC, USA, 2005. IEEE Computer Society.
10. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.
11. M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, Visualizing Objects, Graphs, and Video, pages 83–91, 1992.

An Architecture for Privacy-Aware Inter-domain Identity Management

Wolfgang Hommel

Munich Network Management Team,
Leibniz Supercomputing Center Munich
hommel@lrz.de

Abstract. The management of service oriented architectures demands an efficient control of service users and their authorizations. Similar to structured cabling in LANs, Identity & Access Management systems have proven to be important components of organizations' IT infrastructures. Yet, due to new management challenges such as virtual organizations, on-demand computing and the integration of third party services through composition, identity information has to be passed to external service providers; this decentralization inherently leads to interoperability and privacy issues, which existing management standards are not dealing with appropriately yet. We present an architecture, based on SAML, XACML and XSLT, which provides a tight integration of cross-organizational identity data transfer into the local provisioning business processes along with a policy-driven inter-domain privacy management system, and its implementation.

1 Introduction and Problem Statement

Besides network components and systems, the operation of complex IT infrastructures more and more has to focus on the management of application-level services offered to end users. An essential part of service provisioning is the setup, configuration, maintenance and deletion of user accounts, also known as Identity & Access Management (I&AM). The I&AM paradigm demands to provide a holistic view of a user instead of administrating each account on each service independently. Typically, a central identity repository, such as an LDAP-based enterprise directory, provides the user data required for authentication, authorization and accounting, as well as for service personalization. I&AM systems are usually fed by an organization's human resources (HR) system and customer relationship management (CRM) database; they thus contain sensitive data, which must be protected due to privacy and governance aspects.

However, in an increasing number of scenarios, cross-organizational identity data transfer is required. If, for example, IT services are outsourced to third parties, personalization and accounting data must be made available to the service provider (**SP**), as they are required for service provisioning and billing. Similarly, aligning with other organizations to form a virtual organization, e.g. in Grid projects, requires to pool together parts of the resources and user data

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 48–59, 2005.

alike. To avoid redundant and inconsistent storage of identity information, as well as the administrative overhead to acquire and maintain this data multiple times, dedicated languages and web services based management protocols exist for the exchange of identity information. Standards like the Security Assertion Markup Language (SAML, [1]), the Liberty Alliance specifications [2] and the Web Services Federation Language (WS-Federation, [3]) provide methods which allow an SP to retrieve information about a user from the user's so-called Identity Provider (**IDP**). Especially SAML is in wide-spread use, as it has served as basis for Liberty Alliance and because WS-Federation has adopted SAML support meanwhile. The application of these standards to inter-domain service provisioning is also known as Federated Identity Management (**FIM**).

While those standards provide a lot of much needed inter-domain provisioning functionality, we have shown in previous work that they have several deficiencies in common [4]. In this paper, we present solutions for two of the most urgent problems of the current FIM standards and their existing implementations. First, the demand for an identity federation wide common data schema is not considerate of the syntax and semantics of local I&AM solutions and thus makes the seamless integration of FIM into existing provisioning business processes next to impossible in practice. Second, none of the standards specifies how administrators and users can control and restrict which information about a user is allowed to be sent to which provider, as is urgently required to protect the user's privacy.

We address these issues in this paper by extending the standard SAML architecture by two IDP-side components, while still maintaining full SAML compliance. First, we introduce an Attribute Converter component. It translates incoming requests from the federation-wide data schema into the one used by the local I&AM solution; then, it converts outgoing responses back into the federation's data schema. Second, we demonstrate how the eXtensible Access Control Markup Language (XACML, [5]) can be used efficiently to specify which service providers can access which identity information. We have implemented both components prototypically as extensions to the well-known Shibboleth software [6].

After discussing the state of the art in section 2, we present the concept of our SAML architecture extensions in section 3. We focus on implementational aspects and the introduction of our prototype in section 4 and give an outlook to our further research in section 5.

2 Towards Federated Identity Management

Because more and more services and applications supported the LDAP protocol for both authentication and storage of configuration data, LDAP-based enterprise directories have been widely adopted as basis for intra-organizational I&AM solutions, which focus on the integration and centralized management of an organization's employees, customers and users and their access rights to the local services. Unfortunately, as in many other management areas, no single data schema standard exists, and thus default vendor configurations, such as those found in Microsoft Active Directory or Novell eDirectory, compete with non-proprietary LDAP schema definitions such as inetOrgPerson [7]. In practice, many organizations even create their own LDAP schema to cover their individual needs.

To facilitate cross-organizational identity data exchange, early attempts to grant other organizations access to own enterprise directories quickly turned out to be tedious and suffer from bad scalability. Having to set up accounts for users from other organizations and getting applications to work with different schemas leads to massive administrative overhead and is impractical when more than a handful of organizations is involved.

Thus, dedicated management standards were created, out of which SAML [1] has found wide-spread adopters and is supported by the recent versions of identity management solutions by most big vendors, including HP, IBM, Novell, and Sun. SAML establishes a web services based back channel between the service provider (SP) and the user's home organization, which is called Identity Provider (IDP). Over this back channel, the SP can request information about the user, as shown in figure 1:

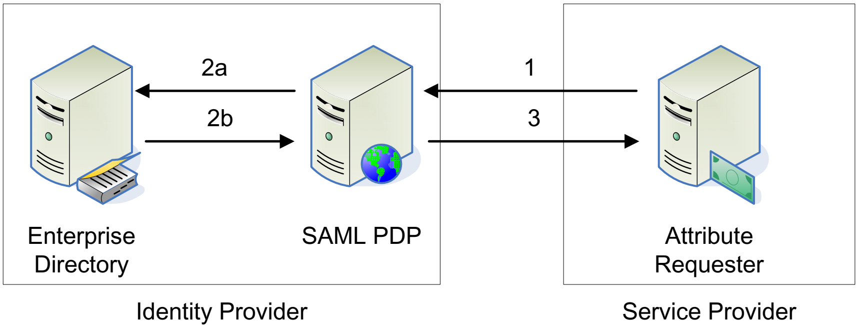


Fig. 1. Identity data exchange through a standard SAML back channel

1. The SP sends a SAML request to the IDP's SAML Policy Decision Point (**PDP**), e.g. it queries the user's billing address. The user is identified by a handle known to both providers, and the billing address consists of *attributes* such as `name`, `street`, `postal code` and `city`, which must have been defined in a federation-wide data schema a priori.
2. Those attributes are looked up in the IDP's local identity repository, which is typically the enterprise directory also used by the local I&AM solution. The result is returned to the IDP's SAML component.
3. The data is wrapped into a SAML attribute assertion and sent back to the SP.

Having to use a federation-wide data schema in a SAML architecture raises two problems: First, finding a common data schema for all involved parties is a non-trivial task due to different technical demands, e.g. different syntactical requirements of applications, and each involved organization's political goals. Second, as an IDP's SAML component must be able to look up the attributes in this schema, the organization either has to use this schema internally as well or provide an extra repository which is synchronized with the local I&AM solution regularly; either way, this causes costs for the extra hardware, synchronization software and operation. Two solution attempts are presently available:

1. The Liberty Alliance provides two standardized schemas, called employee and personal profile ([8], [9]). However, these schemas provide only the greatest common divisor of potentially required identity information, and thus are per se insufficient and have to be extended by other required attributes, similar to their LDAP counterparts.
2. Most vendors support a technique called *attribute mapping*. For example, if an SP requests the `dateOfBirth` attribute, it could be mapped to the `DOB` attribute in the local repository. However, more complex transformations than just renaming an attribute, such as changing the date format from `YYYY-MM-DD` to `DD.MM.YY`, or composing the result from three separate attributes `day`, `month` and `year`, are not possible.

As can also be seen from figure 1, there is no filtering mechanism in place that restricts which attributes are allowed to be sent to the SP. Yet, it is crucial to protect the users' privacy and empower each user to control and restrict which SP has access to which attributes. This issue is also dealt with insufficiently:

- Of the three FIM standards, only Liberty Alliance introduces the idea of *Attribute Release Policies* (**ARP**s). However, it does neither specify the content of such ARPs nor how they should be implemented.
- Only Shibboleth [6], a SAML-based open source FIM software, which is the de-facto standard among higher education institutions, supports ARPs, but in a proprietary format and with rather limited functionality, i.e. the release of each attribute to each SP can be restricted only based on this attribute's current value.

More complex conditions, such as granting access to one's credit card data only if one is actually buying something from a shop and not just browsing for information, cannot be modelled with current ARP concepts and implementations. Also, no obligations can be specified, such as informing a user whenever an SP accesses certain attributes, e.g. by means of an e-mail or a log file.

As both an integration of FIM into existing local business processes and an enhanced privacy protection are urgently required to achieve a smooth setup of identity federations and earning of user acceptance, we have extended the SAML architecture by schema conversion and privacy management components, which are described in the next sections.

3 An Extended SAML Architecture with Schema Conversion and Privacy Management Support

Figure 2 shows our extended SAML architecture; as only the IDP-internal workflow has been modified and the SAML PDP is still the only point of contact to the outside world, we preserve full SAML compliance. We now describe the overall workflow and then go into conceptual details of the attribute conversion component in section 3.1 and specify the privacy management mechanism in section 3.2:

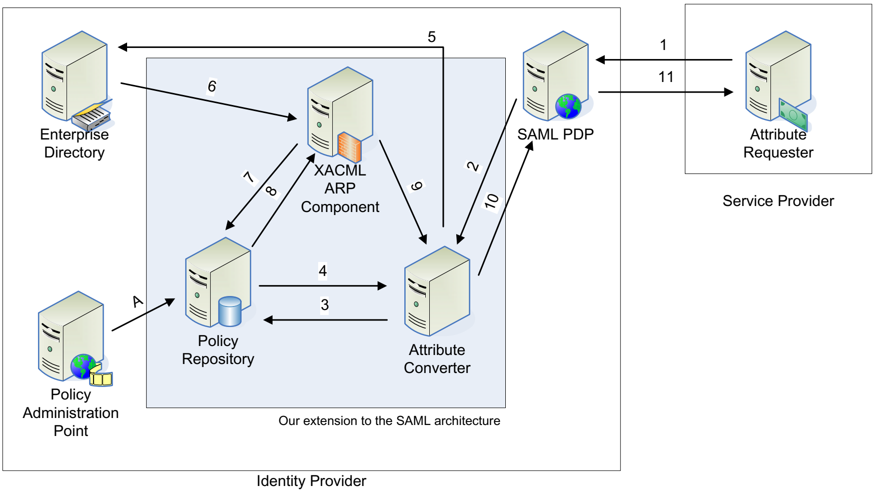


Fig. 2. Extended SAML architecture and workflow

1. The SAML attribute request is sent by the service provider (SP) to the IDP's SAML PDP as before. The data schema used in the request is the federation-wide.
2. The IDP's SAML PDP extracts the list of the wanted attributes from the SAML request; instead of looking them up in the enterprise directory directly, it forwards this list to our attribute converter, which is described in more detail in section 3.1. It also passes information about the requesting SP and the affected user, which are required later to pick the appropriate policies.
3. The attribute converter contacts the IDP's policy repository. It stores pairs of conversion rules which are used to first convert incoming requests from the federation-wide into the locally used data schema, and then convert the results from the locally used to the federation-wide data schema. The rules can be administrated via the policy administration point (see arrow A).
4. The rules required to convert the actually requested attributes to and from the locally used data schema are returned to the attribute converter.

5. The attribute converter translates the list of originally requested attributes into the list of attributes which need to be looked up in the local enterprise directory. By doing so, the names of the attributes can be changed, but attributes may also be added to or deleted from the list, depending on which attributes are required in the local schema to provide the content for the requested attributes in the federation-wide schema. This list of attributes is then looked up in the enterprise directory.
6. Before those attributes can be returned to the SP, Attribute Release Policies (ARPs) are used to protect the user's privacy. We are using the eXtensible Access Control Markup Language (XACML, [5]) to model and enforce ARPs as described in section 3.2. The attributes and their values, which have been retrieved from the enterprise directory, are forwarded to our XACML component.
7. The XACML component looks up the applicable ARPs in the IDP's policy repository. The selection of ARPs depends on various factors, such as the requesting SP, the affected user and the attributes which have been looked up.
8. The relevant ARPs, which typically include IDP-wide ARPs defined by an administrator and user-specific ARPs, are returned to the XACML component. Details are provided in section 4.
9. The XACML ARP component filters the list based on the rules specified by XACML policies as described below and returns only those attributes whose release is allowed back to the attribute converter.
10. This time, the attribute converter has to convert the attributes from the locally used schema back to the federation-wide schema. The necessary rules have already been fetched in step 3. The final result is returned to the IDP's SAML PDP.
11. The IDP's SAML PDP wraps the result in a SAML assertion, which is finally sent to the SP as result of the original attribute request.

The following sections describe the internals of the attribute converter and the XACML ARP component in detail.

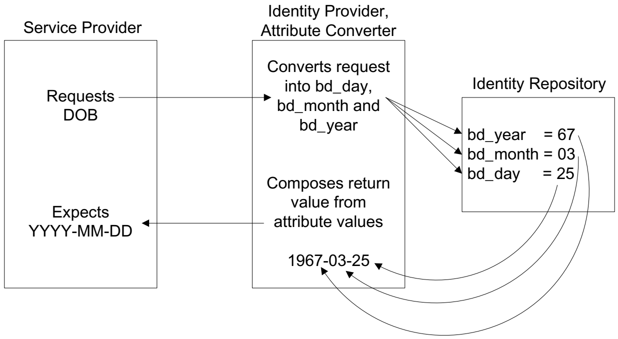


Fig. 3. Solution workflow for a simple schema mismatch example

3.1 Cross-Organizational Identity Schema Conversion

The attribute converter's purpose is to enhance the interoperability of FIM solutions with existing I&AM systems by letting all IDP components work with the IDP's locally established data schema and nevertheless communicate with federation partners transparently.

An example of its use is shown in figure 3. It assumes that a federation has defined a `DOB` attribute which holds the user's date of birth in the format `YYYY-MM-DD`; however, the IDP, which receives the request, stores the user's date of birth in three separate attributes for day, month and year of birth. Even such seemingly simple problems cannot be solved with the existing FIM standards.

The figure also shows the attribute conversion relevant steps of the workflow described above. First, the incoming request for one attribute is modified, so the three locally required attributes are looked up. Then, before returning the result to the SP, the return value for the originally requested attribute is composed from the three separate attributes.

Three kinds of conversions can be made:

1. The names of the requested attributes can be changed, e.g. from `DOB` to `dateOfBirth`. This is equivalent to the attribute mapping approach described in section 2.
2. The attribute's value can be text-processed, in order to fulfill syntactical requirements of the target data schema, or to compose or split up attributes.
3. The attribute's value can be modified to adapt different semantics; for example, the value of a user's `nationality` attribute may have to be `German` in the local schema but `DE` in the federation-wide. Such semantical conversions are eased in practice because many attributes can only have discrete values.

The actual conversion rules are specified as XSLT [10] stylesheets, i.e. XML transformations are performed; an example is given in section 4. XSLT is an obvious choice, as all FIM standards are XML-based and most of their implementations use XML internally as well. Furthermore, also well-established products for local I&AM, such as Novell's Nsure Identity Manager 2, are using XSL transformations for intra-organizational data conversions, so existing programming experience and code can be reused efficiently in the federated case.

3.2 Inter-domain Policy-Based Privacy Management

Privacy management is a well-studied field; standards such as P3P [11] and EPAL [12] have been widely adopted. However, they are intended to specify, publish and enforce privacy policies on the service provider side; they do not specify how the user's preferences shall be stored on the client or identity provider side. Existing implementations, such as Shibboleth [6], use proprietary privacy policy formats on the IDP side; thus, users cannot reuse their policies at other IDPs if they use a different implementation.

We have chosen the eXtensible Access Control Markup Language (XACML, [5]) as basis for the implementation of Attribute Release Policies (ARPs) for the following reasons:

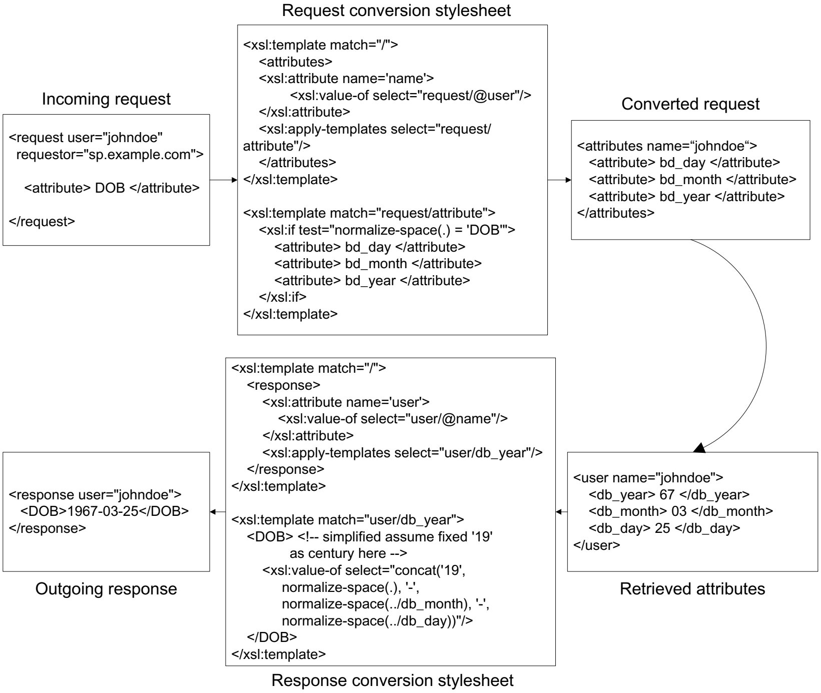
- XACML and SAML have been paired before to achieve fine-grained inter-domain access control, e.g. in well-known systems such as PERMIS [13] and Cardea [14]; a detailed overview can be found in [15].
- XACML is a OASIS standard with a reference implementation available as open source [16]. As XACML is a generic access control language, the policy format can be tailored to individual needs and still be evaluated by any standard compliant XACML PDP; this ensures interoperability and eliminates the need to implement a dedicated PDP. XACML's relationship to P3P, which has been outlined in [17], allows us to leverage a proven privacy standard to FIM applications.
- XACML already provides functionality which is required for advanced ARPs but not available in current proprietary implementations, e.g.
  - Support for multiple roles of a user, e.g. one used at work and one used in spare time.
  - Grouping of attributes, i.e. release rules do not have to be specified for each attribute separately, e.g. as in Shibboleth.
  - Arbitrarily decentralized management, i.e. multiple XACML policies can be combined to form the effective policy. Typically, an IDP administrator will specify default policies which each user can override individually on demand.
  - Very flexible condition formulation; for example, certain attributes may only be released for a certain purpose which the SP has to disclose. Conditions may also contain environmental data such as the current date and time.
  - Formulation of obligations. Logging an SP's access to selected attributes and informing the user by e-mail are two popular obligations which are already part of the XACML standard; arbitrary other obligations can be implemented through XACML's extension mechanisms.
  - Policy protection, i.e. an existing public key infrastructure (PKI) can be used to sign and encrypt the ARPs to prevent unauthorized modification and disclosure.

The XACML PDP component shown in figure 2 consists of an XACML policy enforcement point (PEP) and a standard XACML PDP. The PEP creates XACML requests based on the attributes which are passed in from the enterprise directory (see step 6 of the workflow on page 52). It then fulfills any obligations returned by the XACML PDP and returns the attribute values to the converter component if their release was allowed. An example can be found in the next section.

4 Implementation Details

We will now describe our implementation of the SAML architecture extension and the integration of its components into Shibboleth.

As described in section 3.1, the attribute converter uses XSLT stylesheets to transform incoming requests and outgoing responses. We have implemented this



```
<request user="johndoe"
  requestor="sp.example.com">

   <attribute> DOB </attribute>

</request>
```

```
<xsl:template match="/">
   <attributes>
   <xsl:attribute name='name'>
         <xsl:value-of select="request/@user"/>
   </xsl:attribute>
   <xsl:apply-templates select="request/
attribute"/>
   </attributes>
</xsl:template>

<xsl:template match="request/attribute">
   <xsl:if test="normalize-space(.) = 'DOB'">
      <attribute> bd_day </attribute>
      <attribute> bd_month </attribute>
      <attribute> bd_year </attribute>
   </xsl:if>
</xsl:template>
```

```
<attributes name="johndoe">
   <attribute> bd_day </attribute>
   <attribute> bd_month </attribute>
   <attribute> bd_year </attribute>
</attributes>
```

```
<user name="johndoe">
   <db_year> 67 </db_year>
   <db_month> 03 </db_month>
   <db_day> 25 </db_day>
</user>
```

```
<xsl:template match="/">
   <response>
      <xsl:attribute name='user'>
         <xsl:value-of select="user/@name"/>
      </xsl:attribute>
      <xsl:apply-templates select="user/db_year"/>
   </response>
</xsl:template>

<xsl:template match="user/db_year">
   <DOB> <!-- simplified assume fixed '19'
            as century here -->
      <xsl:value-of select="concat('19',
          normalize-space(.), '-',
          normalize-space(../db_month), '-',
          normalize-space(../db_day))"/>
   </DOB>
</xsl:template>
```

```
<response user="johndoe">
  <DOB>1967-03-25</DOB>
</response>
```

Fig. 4. An example for request and attribute conversions using XSLT

functionality using Xalan [18] as XSLT processor through the standard JAXP Java API. Our prototype uses the local file system as XSLT stylesheet repository; additionally, a web server can be used as policy administration point (PAP) to upload new stylesheets and edit them online (see arrow A in figure 2). Example stylesheets which solve the date of birth schema problem discussed in section 3.1 are shown in figure 4.

Shibboleth provides data connectors for relational databases, LDAP servers and flat text files; it also has an extension mechanism which can be used to hook in custom connectors. By implementing such a custom connector, attribute lookups can be redirected to the attribute converter, which in turn uses the Java JNDI API to retrieve the attributes from the enterprise directory in the local data schema. It passes their values and the meta-data about the service provider, which is delivered by Shibboleth, on to the XACML ARP component.

The XACML ARP component is also implemented in Java, facilitating Sun's XACML PDP implementation [16]. The attribute values and meta-data are received by a custom XACML PEP, which assembles an appropriate XACML request; this request is then evaluated by the PDP, which returns the decision whether the attribute may be released to the given service provider under the given conditions, along with optional obligations. The following example shows an XACML policy which grants access to the user's credit card number to an online shop only if an actual order is placed; an obligation specifies that each allowed release must be logged:

```
 1 <Policy id=”xacmlARP1” RuleCombiningAlg=”first−applicable”>
 2   <CombinerParameters>
 3     <CombinerParameter ParameterName=”ARPpriority”>
 4       100
 5     </CombinerParameter>
 6   </CombinerParameters>
 7   <Description> ARP by user John Doe </Description>
 8   <Rule id=”CreditCardToBookShop” effect=”permit”>
 9     <Description> Release credit card number to bookshop </Description>
10     <Target>
11       <Resources>
12         <Resource>
13           <ResourceMatch MatchId=”string−equal”>
14             <AttributeValue>
15               idp.example.com/johndoe/defaultrole/creditCardNumber
16             </AttributeValue>
17             <ResourceAttributeDesignator AttributeId=”resource−id” />
18           </ResourceMatch>
19         </Resource>
20       </Resources>
21       <Subjects>
22         <Subject>
23           <SubjectMatch MatchId=”string−equal” AttributeValue=”shop.example.com”>
24             <SubjectAttributeDesignator AttributeId=”service_provider” />
25           </SubjectMatch>
26           <SubjectMatch MatchId=”string−equal” AttributeValue=”bookshop”>
27             <SubjectAttributeDesignator AttributeId=”service” />
28           </SubjectMatch>
29           <SubjectMatch MatchId=”string−equal” AttributeValue=”purchase”>
30             <SubjectAttributeDesignator AttributeId=”purpose” />
31           </SubjectMatch>
32         </Subject>
33       </Subjects>
34       <Actions>
35         <Action>
36           <ActionMatch MatchId=”string−equal” AttributeValue=”read”>
37             <ActionAttributeDesignator AttributeId=”action−id” />
38           </ActionMatch>
39         </Action>
40       </Actions>
41     </Target>
42     <Obligations>
43       <Obligation Id=”Log” FulfillOn=”Permit”>
44         <AttributeAssignment Id=”text”>
45           Your credit card number has been released to:
46           <SubjectAttributeDesignator AttributeId=”service_provider” />
47         </AttributeAssignment>
48       </Obligation>
49     </Obligations>
50   </Rule>
51   <Rule id=”DoNotReleaseAnythingElse” effect=”deny”/>
52 </Policy>
```

As can be seen from lines 2–6 of the example, we are using a priority based policy combining algorithm which composes the effective policy out of an arbitrary number of optionally distributed ARPs. In practice, ARPs created through the user's PAP typically have a higher priority than the administrator-specified ARPs, so users can override the IDP's defaults. The attributes to which access is controlled are specified as XACML resources, as shown in lines 11–20 of the example; each attribute is identified globally by its name, which is a URN composed of the IDP identifier, the person, its role and the attribute name as specified in the federation-wide data schema. Three consecutive XACML subject matches control which service provider is actually requesting the attributes for the provisioning of which service and which purpose (see lines 21–33).

An integration into Shibboleth's IDP component can be achieved by first adapting the `listPossibleReleaseAttributes()` method, which must return the names of the user attributes which should be retrieved; second, `filterAttributes()` has to remove all attributes whose release is not permitted by the ARPs. The user's and service provider's ids are passed to both methods, which provides sufficient information for the XACML PEP to identify, combine and let the PDP evaluate the relevant XACML-based ARPs.

Shibboleth's proprietary ARPs can be lossless converted to XACML ARPs. Basically, Shibboleth ARP `targets` become XACML `subjects` and Shibboleth ARP `attribute` elements are converted to XACML `resources`. As release decisions are made on `attribute` and not on `rule` level in Shibboleth ARPs, each Shibboleth `attribute` has to be converted into a dedicated XACML `rule`. We have automated this transformation by also using an XSLT stylesheet.

5 Summary and Outlook

In this paper, we presented a SAML-based architecture for privacy-aware distributed service provisioning, which allows a tight integration of inter-domain provisioning workflows into the individual local identity management business processes. Two urgent problems of current standards have been addressed while still maintaining full compliance. First, we added an attribute converter to the standard SAML architecture; it utilizes XSLT stylesheets to convert incoming SAML attribute requests and outgoing responses from the federation-wide data schema to the locally used one and vice versa, so SAML can be integrated into the local I&AM infrastructure transparently and at minimum cost. Second, to protect each user's privacy across administrative domains, we specified Attribute Release Policies based on XACML, a generic access control language, which has been successfully paired up with SAML for various other purposes before. We demonstrated our implementation and its use on simple real-world problems.

Our further research will focus on improving other weak spots of current FIM standards; in particular, we will study the use of data pushing mechanisms to complement the current pull-only SAML protocol bindings.

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 the paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. The web server of the MNM Team is located at `http://www.mnm-team.org/`.

References

1. Cantor, S., Kemp, J., Philpott, R., Maler, E., (Eds.): Security Assertion Markup Language v2.0. OASIS Security Services Technical Committee Standard (2005)
2. Wason, T., Cantor, S., Hodges, J., Kemp, J., Thompson, P., (Eds.): Liberty Alliance ID-FF Architecture Overview. `http://www.projectliberty.org/resources/specifications.php` (2004)

3. Kaler, C., Nadalin, A., (Eds.): Web Services Federation Language (WS-Federation). http://www-106.ibm.com/developerworks/webservices/library/ws-fed/ (2003)
4. Hommel, W., Reiser, H.: Federated Identity Management: Shortcomings of existing standards. In: Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Management (IM 2005), Nice, France (2005)
5. Moses, T.: OASIS eXtensible Access Control Markup Language 2.0, core specification. OASIS XACML Technical Committee Standard (2005)
6. Cantor, S., Carmody, S., Erdos, M., Hazelton, K., Hoehn, W., Morgan, B.: Shibboleth Architecture, working draft 09. http://shibboleth.internet2.edu/docs/ (2005)
7. Smith, M.: Definition of the inetOrgPerson LDAP Object Class. IETF Proposed Standard, RFC 2798 (2000)
8. Kellomki, S.: Liberty ID-SIS Employee Profile Service Specification. http://project-liberty.org/specs/liberty-idsis-ep-v1.0.pdf (2003)
9. Kellomki, S.: Liberty ID-SIS Personal Profile Service Specification. http://project-liberty.org/specs/liberty-idsis-pp-v1.0.pdf (2003)
10. Clark, J.: XSL Transformations (XSLT), Version 1.0. W3C Recommendation, http://www.w3.org/TR/xslt/ (1999)
11. Reagle, J., Cranor, L.F.: The Platform for Privacy Preferences. In: Communications of the ACM. Volume 42., ACM Press (1999) 48–55
12. Powers, C., Schunter, M.: Enterprise Privacy Authorization Language, W3C member submission. http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/ (2003)
13. Chadwick, D., Otenko, A.: The PERMIS X.509 Role Based Privilege Management Infrastructure. In: Proceedings of the 7th ACM Symposium on Access Control Models and Technologies. SACMAT, ACM Press (2002) 135–140
14. Lepro, R.: Cardea: Dynamic Access Control in Distributed Systems. Technical Report TR NAS–03–020, NASA Advanced Supercomputing Division, Ames (2003)
15. Lorch, M., Proctor, S., Lepro, R., Kafura, D., Shah, S.: First Experiences Using XACML for Access Control in Distributed Systems. In: Proceedings of the ACM Workshop on XML Security, ACM Press (2003)
16. Proctor, S.: Sun's XACML implementation. http://sunxacml.sf.net/ (2004)
17. Anderson, A.H.: The Relationship Between XACML and P3P Privacy Policies. http://research.sun.com/projects/xacml/ (2004)
18. Apache Software Foundation: Xalan XSLT Processor. http://xml.apache.org/xalan-j/ (2005)

Data on Retention

Ward van Wanrooij and Aiko Pras

University of Twente, PO Box 217,
7500 AE, Enschede, The Netherlands
w.a.h.c.vanwanrooij@student.utwente.nl, pras@cs.utwente.nl

Abstract. Proposed EU regulations on data retention could require every provider to keep accounting logs of its customers' Internet usage. Although the technical consequences of these requirements have been investigated by consultancy companies, this paper investigates what this accounting data could be, how it can be obtained and how much data storage is needed. This research shows that every gigabyte of network traffic results in approximately 400 kilobyte of accounting data when using our refinements to existing methods for storing accounting data – less by a factor twenty than previously assumed.

1 Introduction

Recently the discussion regarding the controversial European proposal for a framework decision on the retention of telephone and Internet data [1] stirred emotions across Europe. If enacted this could require every Internet service provider to keep detailed logs of the Internet usage of its customers. Incomprehension about the supposed effectiveness of the proposals and lack of knowledge about the technical implications of the requirements cause an opaque debate dominated by emotions [2] and not by facts.

One major source of confusion appears to be the uneducated guess made by studies regarding the technical consequences [3,4] of the volume and cost of storing so-called "IP accounting data", a term coined in these papers. The amount of and way to store all other relevant traffic data, namely authentication and e-mail logs, is fairly straightforward; however the report estimates that a typical, large access provider needs to store 72 terabytes of IP accounting data yearly. This paper focuses on the definition of this IP accounting data, the way it can be obtained and making an educated guess towards establishing a relation between network data and IP accounting data. Finally, this data is used to establish the expected technical implications for several providers. Note that this paper only documents research into technical aspects of the storage of accounting data and does not give any indication of the effectiveness or efficiency of the proposal and neither approves nor endorses it.

Although much research has been done on traffic analysis, this paper presents a new challenge in storing specific logical connection information while minimizing storage requirements without sampling packets. The resulting process is based on and validated using reliable, real world data. The study "Storage and

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 60–71, 2005.

bandwidth requirements for passive Internet header traces" [5] is based on sampling; the study by KPMG [4] is based on the extrapolation of one unspecified 2 Mbit connection and likewise unspecified "accounting data" to a 25 Gbit load and Cisco [6] specifies a reduction rate for full "netflow data" of 98.5%.

The results of this paper are not only important to law-makers but also to network managers, both policy-makers and administrators, because the enactment of the proposed law can require serious network infrastructure changes. The information in this paper can be used to determine its impact.

The structure of this paper is as follows. Section 2 phrases the research questions, followed by a discussion of the definition of IP accounting data (Sect. 3). Section 4 addresses the way accounting data can be obtained and the results of this process are commented on in Sect. 5. Finally the outcomes of this research are compared to the conclusions of the aforementioned KPMG paper in our conclusion.

2 Research Questions

The three research questions can be summed up as: what is IP accounting data, how can it be obtained and how much storage capacity is needed. More formally, these are the questions:

1. What is, for the purpose of data retention, the description of IP accounting data?
2. In what ways can accounting data be extracted from network traffic?
3. How much accounting data has to be stored ?

3 What Is Accounting Data?

Conceptually, IP accounting data in the context of the data retention initiative has been defined as data pertaining to a connection using "subsets of Internet Protocol numbers" and satisfying one or more of the following criteria [1]:

a. Data necessary to trace and identify the source of a communication which includes personal details, contact information and information identifying services subscribed to.
b. Data necessary to identify the routing and destination of a communication.
c. Data necessary to identify the time and date and duration of a communication.
d. Data necessary to identify the telecommunication.
e. Data necessary to identify the communication device or what purports to be the device.
f. Data necessary to identify the location at the start and throughout the duration of the communication.

Concrete storage proposals based on these requirements for each connection (referred to as communication in the initiative) have been recorded in [7] and include: user, IP addresses, port numbers, date and time and "type of service" (fulfilling conditions a, b and c). For the purpose of this research we assume [3] that all access providers save network authentication logs and consequently are able to map a local IP address to a unique user, based on the combination of local address and time of the connection. We do not expand on this aspect of accounting data in this work.

Now the recordable properties of a connection have been defined, however the definition of a connection has not been. Technically, a connection exists after e.g. a three-way handshake has been made using TCP. For other Internet protocols, like UDP, a similar notion does not exist – therefore a more logical than technical definition has to be used. For this purpose, a modified version of a NetFlow [6] flow has been adapted: *a network flow is defined as a bidirectional stream of packets between a given source and destination within a certain time frame. A flow is defined by its recordable properties.* To avoid confusion, in this document connection refers to a single IP session (TCP connection, UDP message and reply) and flow refers to a stream of packets, possibly spanning multiple connections.

4 How Can Accounting Data Be Obtained?

Several possibilities exist to obtain and store the required accounting data. Three evident options exist:

1. Capturing packet dumps by configuring a special monitoring device on a network link. Because these dumps contain a copy of all network traffic the size is equal to amount of used network bandwidth. Compressing these files results in an average size reduction of 42% [5], nonetheless practically not manageable due to their size.
2. Capturing packet dumps and saving only the first 68 bytes of each Ethernet frame. Depending on the protocol used, these trimmed packets contain at least the full header containing the necessary accounting data and possibly even some payload. A major advantage over plain packet dumps is the reduction in the amount of stored data, compressed a scaling down of up to 90% may be achieved [5]. Such a reduction still results in a 100 MB file for a continuous loaded 2 Mbit/s link for one hour.
3. Saving NetFlow data. Cisco's NetFlow technology, included in many high-end routers and switching devices, generates a flow record (47 bytes) for each unique connection through the network device. The definition of this NetFlow is a superset of the accounting flow definition of Sect. 3. Because the latest incarnation of NetFlow technology has been selected as the basis for IETF's IPFIX (Internet Protocol Flow Information eXtract) [8] and some other vendors[1] already include NetFlow compatible technology in their routers and switches [9], the usage of this technology is a viable choice. In

[1] The competitor sflow can currently not be used for gathering accounting data because sflow is based on sampling.

conjunction with a NetFlow Collector, it allows for realtime network traffic overview on a relatively small scale. However, Cisco's estimate is that the amount of log data is approximately 1.5% of the network traffic and that is undeniably too much for very large scale deployments like the EU data retention initiative.

In the next subsection we introduce a method of processing the input data, be it a (partial) packet dump or NetFlow log, to a format that minimizes long term storage requirements while still conforming to the requirements of accounting data as set forth in Sect. 3. This method extends on the NetFlow data and principles and is referred to as accounting flows for the remainder of this document.

4.1 Accounting Flows Data Definition

First we establish what traffic data we choose to save based on the definition of accounting data and the most efficient way of storing it.

The source port number is typically semi-random and conveys no information so it can be omitted[2] from stored records. The destination port number (when available) expresses important information about the supposed type of service that has been provided (e.g. TCP/80 means HTTP, UDP/53 means DNS). This information is also the only reasonably reliable indicator about the type of service that can be obtained without inspecting, analyzing and possibly decrypting each packet.

Many other NetFlow fields, like number of packets, number of bytes transferred and AS numbers, are irrelevant to this application and can be dropped.

We need to to store two temporal characteristics for each flow: the start and end date. The proposal prescribes a one second resolution for all time related data, but a full timestamp (e.g. seconds since the epoch) requires 4 bytes, each. We opt to instead file the data in units (files, tables, folders) spanning 12 hours of traffic so we can store 2 byte offsets: the offset since the start of the unit for the start date and the duration for the end date.

Summarized, each accounting flow is an instance of this tuple (15 bytes):

Source address. IPv4 address of the host that initiated the flow (4 bytes).
Destination address. IPv4 address of the host that received or rejected the flow (4 bytes).
Destination port number. Port number of this flow on the destination host (2 bytes).
Flow start. Seconds since the epoch when the flow was initiated (2 bytes).
Flow duration. Duration of the flow in seconds (2 bytes).
Protocol. (1 byte) Protocol number used for this flow (e.g. TCP is 6, UDP is 17, GRE is 47).

Further minimization of this data can be obtained by using compression techniques (instead of removing information), e.g. dictionary coding the IP addresses.

[2] In rare cases, the source port number is used for authentication; it can also be used for operation system fingerprinting.

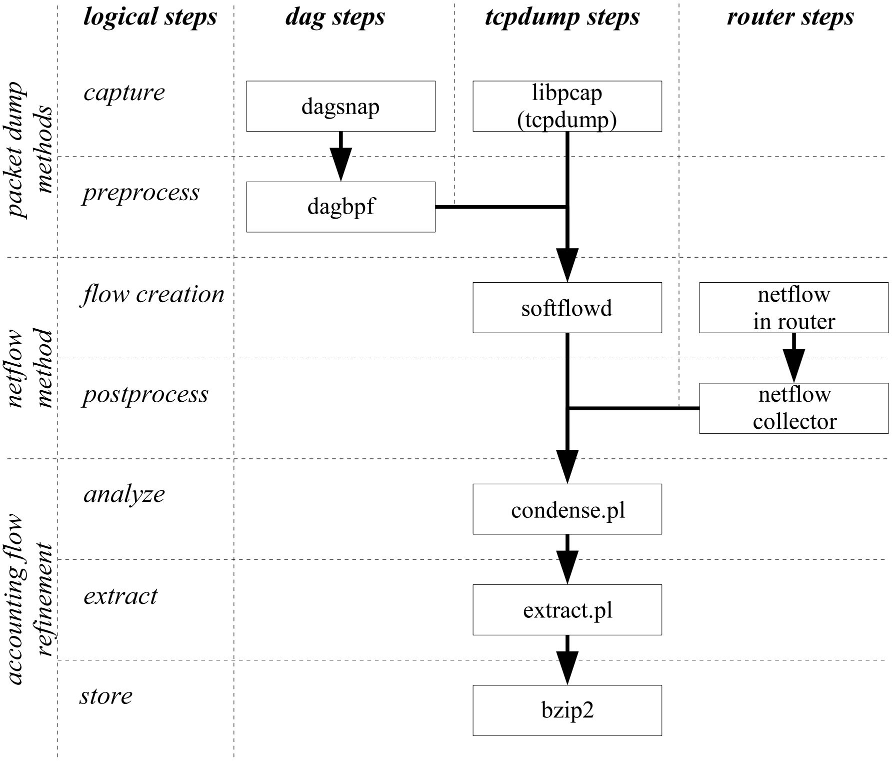


Fig. 1. Steps to obtain accounting flows

No special provisions have been made for IPv6 since its use in connecting end-users is negligible at the moment. However, the above tuple layout is also applicable to IPv6 without changing the size: assuming that the number of connections does not increase when IPv6 is deployed then the total practical number of possible source and destination IP addresses also does not change nor does its storage size. Using a translation table for each unit, an IPv6 address can be mapped to a virtual IPv4 address solely used in the unit for storing the data.

4.2 Obtaining Accounting Flows

Having defined our target data, we need to extend the NetFlow process to output this data and allow two types of input data: realtime data from e.g. NetFlow collectors or a monitoring device and offline data to assist us in testing its performance.

If deployed in production use, the input data will most probably be NetFlow logs because this is more efficient, scalable and reliable than using full or partial packet traces.

The procedure to go from network traffic to accounting flows can be characterized as a seven step process (Fig. 1):

Capture. The action of capturing the packets, we use dagsnap and libpcap based tools.

Preprocess. Processing the captured data to provide suitable input to the next step (e.g. collecting, assembling, converting). In this paper, data captured using DAG [12] cards needs to be converted to libpcap format.

Flow creation. Creating flows from the preprocessed data. The algorithm has been specified in [6]: in essence NetFlow groups related packets together in a flow (logical connection) based on, among other things, IP addresses and port numbers. After the end of the connection or a certain period of inactivity the flow is expired and exported. We utilized the package softflowd[3] [10] for the conversion of libpcap dumps. The parameters for flow expiration are 5 minutes of inactivity on a flow or a maximum life of 12 hours or a maximum traffic of 2 gigabytes. NetFlow enabled routers natively export NetFlow data and combines this and the previous two steps.

Post process. (optional) Processing the flows to provide storable output. When using NetFlow, the flows are gathered by a NetFlow collector in this step.

Analyze. Transforming the netflows into accounting flows. For this paper the step is achieved by running the data through a custom Perl program that removes and regroups data based on the data definition. Its main effect is achieved by removing the source port and grouping tuples of equal properties (source address, destination address, destination port, protocol) together while still considering the other (time, space) constraints.

Extract. Extracting the relevant data from the analyzed data and saving it in binary format. We use a simple script to perform these tasks.

Store. Storing the extracted accounting flows in compressed form for archival purposes. For this evaluation we make use of the general purpose program bzip2 [11] to pack the data.

5 How Much Accounting Data Has To Be Stored?

To accurately measure the storage requirements to save the accounting data and substantiate the refined method we need representative input data of several types of locations. Further, this input data needs to be available in packet dumps and not NetFlow logs because dumps offer greater insight during development and are more prevalent than NetFlow logs. For our research we have selected over 30 traces from six different locations (table 1), together representative for most public networks.

These six locations can be classified into three categories of samplepoints:

1. End-users network uplink: m2c-loc4 monitors basic broadband (ADSL, varying speeds from 256 Kbit/s to 8 Mbit/s) connections, m2c-loc1 captures

[3] Several patches (file output, report aggregates, distinguish between source and destination of a flow) have been made to enhance softflowd. The complete set is available at http://wwwhome.cs.utwente.nl/~wanrooij/papers/dataonretention/-softflowd.patch

Table 1. Samplepoints

name	type	date	#traces	#days	time (hr:mn)	traffic (GB)
m2c-loc1 [13]	student dorms	2002Q2	4	4	01:00	51
m2c-loc3 [13]	college	2003Q3	8	5	02:00	16
m2c-loc4 [13]	broadband homes	2004Q1	15	8	03:45	184
sigcomm-01 [14]	conference	2001Q3	1	3	52:00	4
nzix-II [15]	Internet exchange	2000Q3	3	3	14:30	37
mawi-b [16]	transatlantic link	2005Q1	12	1	03:00	23

packets from heavy broadband users (Ethernet, 100 Mbit/s), m2c-loc3 is a college allowing 9 to 5 usage of its network by students (comparable to an office setting) and sigcomm-01 is a trace of a wireless 802.11b conference network.

2. Inter-network point: nzix-II is a set of traces of the NZIX when it served as a peering point for six major New Zealand ISPs. These files, captured by a DAG [12] card, need to be converted to libpcap format before use.
3. Intra-network link: mawi-b is a transpacific (Japan - United States) 100 Mbit/s line.

The last category of intra-network link serves as verification whether possible conclusions about the amount of accounting data might also be applicable to other links.

5.1 Results

For each of the categories of Sect. 5 the amount of stored date for each of the methods in Sect. 4 and the new accounting flows refinement has been determined using the steps of Sect. 4.2. These amounts have been visualized in figures 2 and 3; when applicable, charts have separate data series for different times of the day. Along the X axis in Fig. 2 are the different methods: full dump (traffic), header dump (header), netflow, accounting flows (accflow) and the compressed version of accounting flows (bzip2). Finally each graph contains an indicator on the optimality of the accounting flow algorithm: this calculated value is based on the principle that every combination of source and destination address should at least appear once in the flows. Each combination of two addresses that appears more than once in the accounting flows, e.g. due to different port number or expiration, is only present once in this indicator.

Table 2 shows the ratio (compressed accounting flows)/(traffic data) for each of the locations.

The four end-user locations share the common impression of a steep descending line in the graphs, but some variation in the resulting ratios still exist. The differences can be explained by looking at the usage of the connection at the respective location:

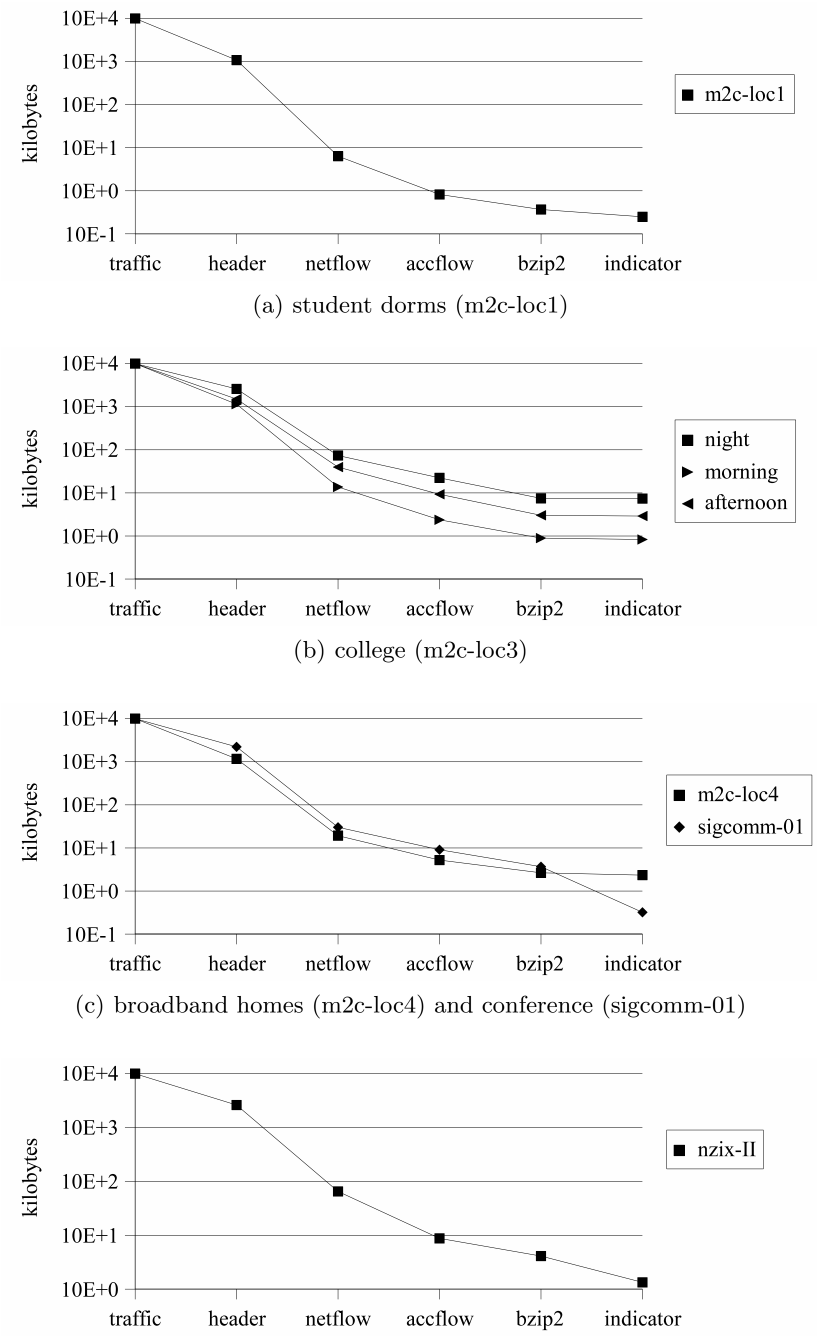
Table 2. Unweighed ratio compressed accounting data/traffic

name	category	ratio	name	category	ratio
m2c-loc1	end-user	0.00367%	nzix-II	inter-network	0.04116%
m2c-loc3-night	end-user	0.07485%	sigcomm-01	end-user	0.03670%
m2c-loc3-morning	end-user	0.00892%	mawi-b-03h	intra-network	0.17977&
m2c-loc3-afternoon	end-user	0.03019%	mawi-b-15h	intra-network	0.26824%
m2c-loc4	end-user	0.02646%	mawi-b-21h	intra-network	0.01137%

– Student dorms (Fig. 2(a)): Probably the most notable aspect are the small ratios for m2c-loc1 in comparison with all other locations. The explanation for this phenomenon is straightforward: when connections are stable (same characteristics) and relatively large amounts of data are transferred using these connections then a smaller set of flows is produced than when the connections are used for e.g. browsing or messaging. This location connects student dorms using 100 Mbit/s endpoints: an adequate interpretation of this data, also glancing over the used port numbers and direction of connections, is up and downloading of large files (sharing).
– College (Fig. 2(b)): The traffic dump for this location has been preprocessed before flow creation was attempted due to large inexplicable amounts of ICMP traffic. Because "normal" traffic data does not exhibit this pattern this ICMP data has been filtered out; all figures (e.g. amount of traffic data) have also been corrected for this filtering. The diverging ratios for different times of the day are striking. However, considering the type of location (college, no student dorms) and the observations of m2c-loc1 the explanation is obvious: at night the Internet traffic is limited to e.g. DNS lookups, some mails and some background traffic (all activities resulting in very high ratios); in the morning students come in and start sharing files (low ratio), most of them already left the building by the end of the afternoon (higher ratio).
– Broadband homes (Fig. 2(c)) and sigcomm-01 (Fig. 2(c)) represent typical web activities and share similar results. Sigcomm being a conference engages in more casual, quick browsing and net accessing while m2c-loc4 (ADSL network) has more transfer of files.

The inter-network point shows a declining curve for nzix-II (Fig. 2(d)) that approximately matches the results of the end-users connections. The free fall in the graph near the indicator is caused by the fact that the traces used cover a period of six hours, so expired flows, e.g. POP3 sessions and news websites, are more prevalent than in a short trace. This also applies to the sigcomm-01 trace of several days.

The intra-network samplepoint differs from the others in it being an arbitrary link in a network that connects two large IP networks in different time zones. Because we used parts of a 24 hour long trace and initial analysis revealed wildly varying traffic patterns throughout the day, the graph has been set-up in a different way: the x-axis is now time measurement and the data series are the amount of data of the respective steps (Fig. 3). Because the link primarily



(a) student dorms (m2c-loc1)

(b) college (m2c-loc3)

(c) broadband homes (m2c-loc4) and conference (sigcomm-01)

(d) internet exchange (nzix-II)

Fig. 2. Accounting data per 100 MB traffic

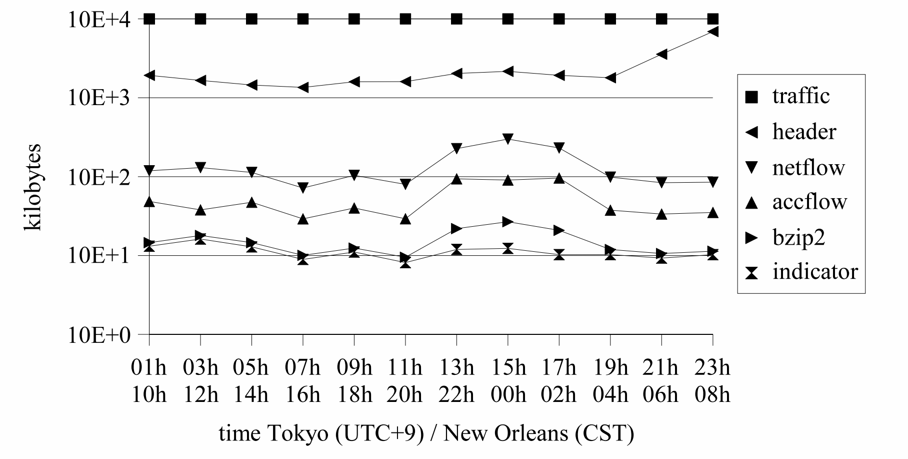


Fig. 3. Accounting data for mawi-b per 100 MB traffic

connects Japan and USA, time has been marked in local (Tokyo) and Central Standard Time. Upon investigation (destination port, traffic patterns), a lot of virus related activity seems to be going on on the link (scans for targets by worms), especially during the spikes around 03:00 and 16:00. Now the value of the two timelines comes around: when intentional traffic is only unidirectional, because the other side is mostly asleep, traffic can become distorted because worms and viruses don't ever sleep. These malignant programs usually scan complete networks just by sending a few UDP or TCP packets, thereby creating a different flow entry for each target, hence the amount of accounting flows substantially increases per 100 MB traffic. The fact that the second spike is higher than the first one does not conclusively prove anything; this may be caused by:

- Larger IPv4 network in Japan reachable through the link (more targets: more traffic).
- Larger IPv4 network in USA reachable through the link (more perpetrators: more traffic).
- Larger percentage of infected workstations in USA
- Different browsing habits in Japan and USA

Aside from these interesting observations, the traffic patterns, although always distorted by worms due to the nature of the link, confirm that our other observations are approximately on target, even though these results are higher. This is easily explained due to the nature of the link: bidirectional, not only connecting end users to servers but also servers and worms to end users.

5.2 Comparison

Based on the results in tables 1 and 2, we can make the cautious, but reasonable estimate for the ratio (compressed accounting data in bytes)/(network traffic in bytes) of 0.04%. This figure is applicable to both traffic on network uplinks as well as traffic on network exchanges; graph 3 established that this figure could also be applied to other links, except for special circumstances.

These results differ from previously published findings by KPMG. This report [4] asserted that one provider stated that one hour of 2 Mbit/s traffic generates 8 megabyte accounting data (ratio of 0.89%). Based on the average throughput of AMSIX [17], the major Internet exchange in the Netherlands, of 25 Gbit/s, an estimate of 60 terabyte accounting data per month (0.76%) is reasoned out in the report. Although not specified in the report, we assume that the format for saving data is compressed NetFlow.

Using our refinements to the NetFlow technology for storing accounting flows requires just 0.04% disk space of the original traffic, a twenty fold increase in efficiency. Instead of 60 terabyte of storage, now only an estimated 3 terabyte is required monthly to store all the logs.

6 Conclusion

Motivated by the confusion surrounding the storage requirements of the possible EU regulations on data retention [1], we have investigated the definition of accounting data and ways to extract this data from network traffic.

During our research we have refined the NetFlow method to allow for efficient storage of the information required by the data retention proposal. The testing of these enhancements on several representative data sets shows a reduction in storage data of about 99.96% – equal to approximately 400 kilobyte of accounting flows for every gigabyte of bandwidth used. Previous research into this area by the consultancy company KPMG [4] documents a ratio of about 0.8% percent; this specialized method performs better by a factor twenty. For the Netherlands this represents a reduction of 57 terabyte of total monthly storage needed to comply with the data retention initiative.

The results of this research can and should be used to make the debate on the EU data retention laws more transparent and factually correct. Because the findings on the definition, process and ratios are universal, they can also be used in any debate on or application of data retention – whether in public law by policy makers or private venue by network managers.

References

1. Presidency of Council of the European Union: Draft Framework Decision on the retention of data processed and stored in connection with the provision of publicly available electronic communications services or data on public communications networks for the purpose of prevention, investigation, detection and prosecution of crime and criminal offences including terrorism, Brussels, Belgium, November 2004, http://register.consilium.eu.int/pdf/en/04/st14/st14190.en04.pdf
2. Persson, M., Trommelen, J.: Ten aanval, Volkskrant 12 April 2005, PCM Uitgevers, Amsterdam, The Netherlands, April 2005

3. Stratix: Onderzoek “Bewaren Verkeersgegevens door Telecommunicatieaanbieders”, Schiphol, The Netherlands, August 2003, http://www.bof.nl/docs/-stratix_verkeersgegevens_eindrapport.pdf
4. KPMG Information Risk Management: Onderzoek naar de opslag van historische verkeersgegevens van telecommunicatieaanbieders, Amstelveen, The Netherlands, November 2004, http://www.bof.nl/docs/bewaarplicht_KPMG.pdf
5. Micheel, J., Braun, H.-W., Graham, I.: Storage and Bandwidth Requirements for Passive Internet Header Traces, Workshop on Network-Related Data Management, in conjunction with ACM SIGMOD/PODS 2001, Santa Barbara, California, USA, May 2001, http://moat.nlanr.net/Papers/nrdm2001.pdf
6. Cisco: NetFlow Services Solutions Guide, San Jose, USA, October 2001, http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite.pdf
7. Working party on co-operation on criminal matters: Non paper data retention, Leiden, The Netherlands, September 2004, http://www.bof.nl/docs/non-paper.pdf
8. IETF Secretariat: IP Flow Information Export (ipfix) Charter, May 2005, http://www.ietf.org/html.charters/ipfix-charter.html
9. Kretchmar, J.: Open Source Network Administration, Prentice Hall PTR, Upper Sadle River, New Jersey, USA, September 2003, Section 5.1
10. Miller, D.: Software NetFlow probe, May 2005, http://www.mindrot.org/-softflowd.html
11. Seward, J.: bzip2, May 2005, http://www.bzip.org/
12. Endace Measurement Systems: Network Monitoring Cards, May 2005, http://-www.endace.com/networkMCards.htm
13. Van de Meent, R.: M2C Measurement Data Repository, Enschede, The Netherlands, December 2003, http://arch.cs.utwente.nl/projects/m2c/m2c-D15.pdf
14. Balachandran, A.: Wireless LAN Traces from ACM SIGCOMM’01, San Diego, California, USA, August 2001, http://ramp.ucsd.edu/pawn/sigcomm-trace/
15. WAND Research Group: NLANR MOAT NZIX-II trace archive, May 2005, http://pma.nlanr.net/Traces/long/nzix2.html
16. WIDE MAWI Working Group: MAWI Working Group Traffic Archive, May 2005, http://tracer.csl.sony.co.jp/mawi/
17. AMS-IX B.V.: AMS-IX Homepage, May 2005, http://www.ams-ix.net

SLA Design from a Business Perspective

Jacques Sauvé[1], Filipe Marques[1], Antão Moura[1], Marcus Sampaio[1],
João Jornada[2], and Eduardo Radziuk[2]

[1] Universidade Federal de Campina Grande, Brazil
{jacques, filipetm, antao, sampaio}@dsc.ufcg.edu.br
[2] Hewlett-Packard-Brazil
{joao.jornada, eduardo.radziuk}@hp.com

Abstract. A method is proposed whereby values for Service Level Objectives (SLOs) of an SLA can be chosen to reduce the sum IT infrastructure cost plus business financial loss. Business considerations are brought into the model by including the business losses sustained when IT components fail or performance is degraded. To this end, an impact model is fully developed in the paper. A numerical example consisting of an e-commerce business process using an IT service dependent on three infrastructure tiers (web tier, application tier, database tier) is used to show that the resulting choice of SLOs can be vastly superior to ad hoc design. A further conclusion is that infrastructure design and the resulting SLOs can be quite dependent on the "importance" of the business processes (BPs) being serviced: higher-revenue BPs deserve better infrastructure and the method presented shows exactly how much better the infrastructure should be.

1 Introduction

Service Level Agreements (SLAs) are now commonly used to capture the performance requirements that business considerations make on information technology (IT) services. This is done both for services provided in-house and for outsourced services. An SLA defines certain Service Level Indicators (SLIs) and restrictions that such indicators should obey. Restrictions are frequently expressed in the form of Service Level Objectives (SLOs), threshold values that limit the value of SLIs. Some typical SLIs are service availability, service response time, and transaction throughput. The problem examined in this paper is that of designing SLAs; the SLA design problem is informally defined as that of choosing appropriate values for SLOs. For example, should service availability be 99.9%, 99.97%? How is one to choose adequate values? There are other aspects to SLA design (choosing SLIs, choosing measurement methods and periods, choosing penalties, etc.) but these are not considered here.

It is interesting to examine how choosing SLOs is typically done today. Naturally, since SLOs are chosen according to how important a service is to the business, the IT client (a senior business manager) is involved in choosing SLOs. However, as reference [11] has vigorously shown, the methods used are almost

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 72–83, 2005.

always pure guesswork, frequently resulting in drastic loss or penalties. It is clear that one needs more mature and objective models to properly design SLAs. An approach based on Business Impact Management [3,12] is presented in this paper.

The remainder of the paper is organized as follows: section 2 informally discusses the approach while section 3 formalizes it; section 4 considers an application of the method through a full numerical example; section 5 discusses related work; conclusions are provided in section 6.

2 Gaining a Business Perspective on IT Operations

An informal discussion of the approach adopted here will help the reader follow the formal treatment presented in the next section.

2.1 Addressing IT Problems Through Business Impact Management

SLOs must be chosen by taking into account the importance of the IT service on the business. In the approach being described here, this is done by capturing the impact of IT faults and performance degradations on numerical business metrics associated with the business. By considering business metrics, one may say that the approach is part of a new area of IT management called Business Impact Management (BIM) [3,12]. BIM takes Service Management (SM) to a new maturity level since metrics meaningful to the customer such as financial or risk measures are used to gauge IT effectiveness rather than technical metrics such as availability and response time.

For BIM to be successfully applied to the problem at hand, one needs to construct an impact model. Since it is quite difficult to bridge the gap between events – such as outages – occurring in the IT infrastructure and their financial effect on the business, an intermediate level is considered: that of the business processes (BPs) using the IT services. Thus, an impact model is used to map technical service metrics to BP metrics such as BP throughput (in transactions per second) and a revenue model to map BP throughput to a final business metric such as revenue throughput.

Thus, this paper essentially investigates how BIM can be useful in addressing some common IT problems. SLA design was chosen as an example of an activity performed by IT personnel that can be rethought from a business perspective using BIM.

2.2 SLA Design: An Optimization Problem

The IT infrastructure used to provision IT services is designed to provide particular service levels and these are captured in SLAs. Intuitively, a weak infrastructure (with little redundancy or over-utilized resources) has the advantages of having low cost but may generate high business losses – as captured by the BIM impact model – resulting from low availability and customer defections due to high response times. An infrastructure with much better availability and lower response times will possibly generate lower business losses but may have a much higher total cost of ownership (TCO). Thus, in both cases, total financial outlay (TCO plus business losses) may be high. It thus appears that a middle ground can be found that will minimize this sum. Once this infrastructure yielding minimal financial outlay is found, one may then calculate SLOs such as availability and response time. As a result, SLO thresholds will be *outputs* from the method rather than being chosen in an ad hoc way. These SLOs will be optimal in the sense that they will minimize total financial outlay.

3 Problem Formalization

The optimization problem considered aims to calculate the number of load-balanced resources and the number of fail-over resources to be used in provisioning IT services so as to minimize overall cost (TCO plus business losses). The model considers workloads with fixed averages and static resource allocation. Once this infrastructure is found, SLOs such as service availability, average response time, etc. can be calculated and inserted in the SLA. This section formalizes the SLA Design problem.

3.1 The Entities and Their Relationships

Figure 1 shows the entities and their relationships used in the problem formalization. It can also be useful to the reader as a quick reference to the notation employed. The model includes entities both from the IT world and the business world. The business (top) layer consists of several business processes. For simplicity, assume that there is a one-to-one relationship between business processes and IT services. Extension to several services is straightforward but would needlessly complicate the formalism for this presentation. We thus have a set BP of BPs and a set S of services: $S = \{s_1, \ldots, s_{|S|}\}$. The infrastructure used to provision these services consists of a set RC of resource classes.

Service s_i depends upon a set RC_i^S of these resource classes. For example, a service could depend on three resource classes: a Web resource class, an application server resource class and a database resource class. Class RC_j consists of a cluster of IT resources. This cluster has a total of n_j identical individual resources, up to m_j of which are load-balanced and are used to provide adequate processing power to handle incoming load. The resources that are not used in a load-balanced cluster are available in standby (fail-over) mode to improve availability.

Finally, an individual resource $R_j \in RC_j$ consists of a set $P = \{P_{j,1}, \ldots, P_{j,k}, \ldots)$ of components, all of which must be operational for the resource to also be operational. As an example, a single Web server could be made up of the following components: server hardware, operating system software and Web server software. Individual components are subject to faults as will be described later.

An SLA is to be negotiated concerning these services. For service s_i, the SLA may specify Service Level Objectives (SLOs). The impact model to be presented assumes that BP throughput is lost if the service is unavailable or

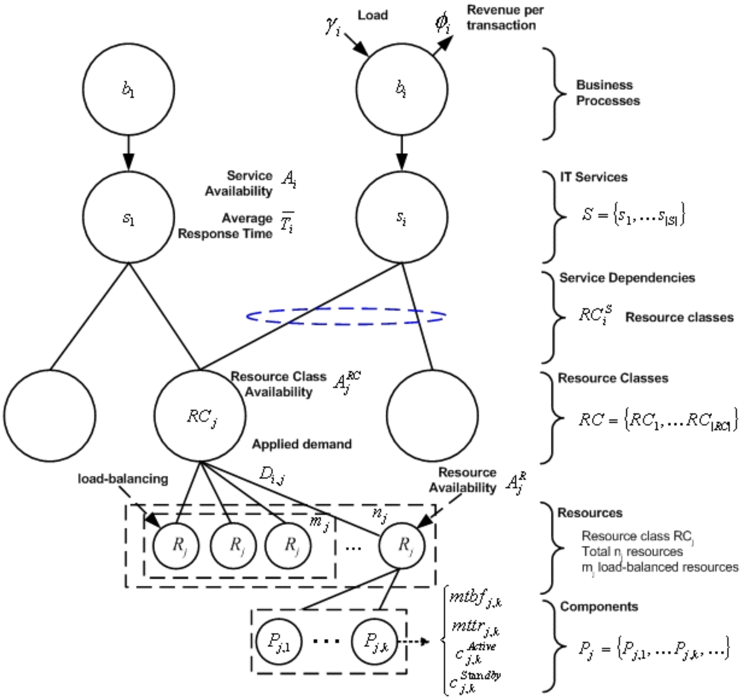


Fig. 1. Entities and their relationships

if response time exceeds a certain threshold. The following SLO parameters are considered for service s_i and will constitute the promise made to the customer in the SLA: A_i^{MIN}, the minimum service availability, $\bar{T}_i$, the average response time, T_i^{DEF}, the response time threshold causing customer defection and $B_i\left(T_i^{DEF}\right) = B_i^{MAX}$, the probability that response time is larger than the threshold.

One may thus summarize the SLA as the four sets: $A^{MIN} = \{\ldots, A_i^{MIN}, \ldots\}$, $T = \{\ldots, \bar{T}_i, \ldots\}$, $T^{DEF} = \{\ldots, T_i^{DEF}, \ldots\}$, $B^{MAX} = \{\ldots, B_i^{MAX}, \ldots\}$.

3.2 The Cost Model

Each infrastructure component $P_{j,k}$ has a cost rate $c_{j,k}^{Active}$ when active (that is, used in a load-balanced server) and has a cost rate $c_{j,k}^{Standby}$ when on standby. These values are cost per unit time for the component and may be calculated as its total cost of ownership (TCO) divided by the amortization period for the component. The cost of the infrastructure over a time period of duration ΔT can be calculated as the sum of individual cost for all components. In the equation below, j runs over resource classes, l runs over resources and k runs over components.

$$C\left(\Delta T\right) = \Delta T \cdot \sum_{j=1}^{|RC|} \left(\sum_{l=1}^{m_j} \sum_{k=1}^{|P_j|} c_{j,k}^{Active} + \sum_{l=1}^{n_j - m_j} \sum_{k=1}^{|P_j|} c_{j,k}^{Standby} \right) \tag{1}$$

3.3 Loss Considerations

A weak infrastructure costs little but may generate large financial losses due to low availability or high response time. The converse situation is an infrastructure that causes little loss but is expensive to provision. In order to evaluate this tradeoff, financial loss must be calculated. In general, the model used is that at time t, the imperfect infrastructure produces adverse impact on business – or simply business loss – at rate $l\left(t\right)$; the rate is expressed in units appropriate to the business metric used per time unit. As an example, loss rate could be expressed in dollars per second when using dollar revenue as a business metric.

For simplicity, assume that all SLOs are evaluated at the same time and that the evaluation period is ΔT. Thus, the accumulated business impact over the evaluation period is $L\left(\Delta T\right) = \int_0^{\Delta T} l\left(t\right) dt$. Assuming a constant rate (l) of faults over time, we have $L\left(\Delta T\right) = \Delta T \cdot l$. A specific loss model will be discussed below.

3.4 The SLA Design Problem

The SLA Design problem may be stated informally as follows: one wishes to determine the number of servers – both total number of servers and number of load-balanced servers – that will minimize the financial impact on the enterprise coming from two sources: infrastructure cost and financial loss. Formally, a first SLA Design problem may be posed as follows:

Find:	The SLA parameters, the sets A^{MIN}, T, B^{MAX}
By minimizing:	$C(\Delta T) + L(\Delta T)$, the total financial impact on the business over evaluation period ΔT
Over:	$\left\{n_1, \ldots, n_{\left\|RC\right\|}\right\}$ and $\left\{m_1, \ldots, m_{\left\|RC\right\|}\right\}$
Subject to:	$n_j \geq m_j$ and $m_j \geq 1$
Where:	$C(\Delta T)$ is the infrastructure cost over the SLA evaluation period ΔT; $L(\Delta T)$ is the financial loss over the SLA evaluation period ΔT; n_j is the number of resources in resource class RC_j; m_j is the number of load-balanced resources in RC_j.

The set $T^{DEF} = \left\{\ldots, T_i^{DEF}, \ldots\right\}$ which indicates the response time threshold from which defections start to occur is given as input. A typical value is 8 seconds for web-based e-commerce [13]. As a result of the optimization, values for the three sets of SLA thresholds availability: $A^{MIN} = \left\{\ldots, A_i^{MIN}, \ldots\right\}$, average response time: $T = \left\{\ldots, \bar{T}_i, \ldots\right\}$, and defection probability: $B^{MAX} = \left\{\ldots, B_i^{MAX}, \ldots\right\}$ will be found. These are the values to be used in an SLA.

In order to complete the model, one needs to define an impact model and a way to calculate loss $L(\Delta T)$, and the SLOs A^{MIN}, T, and B^{MAX}. The next sections cover this.

3.5 A Specific Loss Model

When IT problems occur, the impact on business may be decreased revenue or increased costs or both. In this paper only decreased revenue is considered, a situation applicable to revenue-generating BPs typical in e-commerce. Each BP has an input load (in transactions per second). Some of this load is lost due to a loss mechanism with 2 causes: service unavailability and customer defection due to high response times. Subtracting lost load from the input load results in the BP transaction throughput (denoted by X). The revenue throughput due to any given business process is $V = X \cdot \phi$ where ϕ is the average revenue per transaction for the business process. The total loss rate, over all BPs is

$$l = \sum_{i=1}^{|BP|} l_i$$

where BP is the set of BPs and l_i is the loss rate due to BP b_i. In the above, we have $l_i = \Delta X_i \cdot \phi_i$. Here, ΔX_i is the loss in throughput (in transactions per second) for BP b_i and ϕ_i is the average revenue per transaction for process b_i.

We consider that the BP is heavily dependent on IT, and thus BP availability A_i is equivalent to the availability of the IT service (s_i) used by the BP. When service s_i is unavailable, throughput loss is total and this occurs with probability $1-A_i$. We thus have $\Delta X_i^A = \gamma_i \cdot (1 - A_i)$ where ΔX_i^A is loss attributable to service unavailability, γ_i is the input load incident on BP b_i and A_i is the availability of service s_i. When service is available (this occurs with probability A_i), loss occurs when response time is slow. Thus, we have $\Delta X_i^T = \gamma_i \cdot B_i \left(T_i^{DEF}\right) \cdot A_i$ where ΔX_i^T is loss attributable to high response time, $B_i \left(T_i^{DEF}\right) = Pr \left[\tilde{T}_i > T_i^{DEF}\right]$ is the probability that the service response time (the random variable $\tilde{T}_i$) is larger that some threshold T_i^{DEF}. This models customer defection and assumes that a customer will always defect if response time is greater than the threshold (typically 8 seconds for an e-commerce BP).

The total loss in BP throughput is simply the sum of losses due to unavailability and losses due to high response time:

$$\Delta X_i = \Delta X_i^A + \Delta X_i^T = \gamma_i \cdot (1 - A_i) + \gamma_i \cdot B_i \left(T_i^{DEF}\right) \cdot A_i \tag{2}$$

3.6 The Availability Model

In order to calculate lost throughput, one needs to evaluate the availability A_i of an IT service, s_i. This is done using standard reliability theory [15]. Individual component availability may be found from Mean-Time-Between-Failures (MTBF) and Mean-Time-To-Repair (MTTR) values. Since all components must be available for a resource to be available, the component availabilities are combined using "series system reliability" to yield resource availability A_j^R. Combining resource availability to compute resource class availability (A_j^{RC}) uses "m-out-of-n reliability" since the resource class will be available and able to handle the projected load when at least m_j resources are available for load-balancing. Finally, for service s_i to be available, all resource classes it uses must be available and "series system reliability" is used to calculate service availability (A_i).

3.7 The Response Time Performance Model

The loss calculation depends on $B_i\left(T_i^{DEF}\right)$, the probability that the service response time is larger that some threshold T_i^{DEF}. In order to find this probability, the IT services are modeled using an open queuing model. This is adequate for the case of a large number of potential customers, a common situation for e-commerce. Each resource class RC_j consists of a cluster of n_j resources, of which m_j are load-balanced. Let us examine service s_i. The input rate is γ_i transactions per second. Each transaction demands service from all resource classes in the set RC_i^S. Demand applied by each transaction from BP b_i on class RC_j is assumed to be $D_{i,j}$ seconds. In fact this is the service demand if a "standard" processing resource is used in the class RC_j resources. In order to handle the case of more powerful hardware, assume that a resource in class RC_j has a processing speedup of α_j compared to the standard resource. Thus, service time for a transaction is $D_{i,j}/\alpha_j$ and the service rate at a class RC_j resource for transactions from business process b_i is $\mu_{i,j} = \alpha_j/D_{i,j}$. Finally, since there are m_j identical load-balanced parallel servers used for processing in resource class RC_j, response time is calculated for an equivalent single server [13] with input load $\lambda_{i,j} = \gamma_i/m_j$. Thus the utilization $\rho_{i,j}$ of class RC_j resources in processing transactions from business process b_i is:

$$\rho_{i,j} = \frac{\lambda_{i,j}}{\mu_{i,j}} = \frac{\gamma_i \cdot D_{i,j}}{m_j \cdot \alpha_j} \tag{3}$$

The total utilization ρ_j of class RC_j resources due to transactions from all services is:

$$\rho_j = \sum_{i=1}^{|S|} \rho_{i,j} \tag{4}$$

Observe that, when load is so large that any $\rho_j \geq 1$, then any service depending on that resource class will have $B_i\left(T_i^{DEF}\right) = 1$, since response time is very high for saturated resources.

Now, in order to find $B_i\left(T_i^{DEF}\right)$ when $\rho_j < 1$, let us find the cumulative distribution of response time, $T_i(y) = Pr\left[\tilde{T}_i \leq y\right]$. In this case, the total response time for a transaction from BP b_i is the sum of $\left|RC_i^S\right|$ random variables, one for each resource class used by service s_i. In order to find the probability distribution of a sum of independent random variables, one may multiply their Laplace transforms [14]. In order to make mathematical treatment feasible, assume Poisson arrivals (this is a reasonable assumption for stochastic processes with large population) and exponentially distributed service times. (Observe that although service times may not be independent and exponentially distributed in practice, the optimization step *compares* design alternatives and that is probably insensitive to particular distributions – if they are the same when comparing results.) From queuing theory, the Laplace transform of response time (waiting time plus service time) for a single-server queue is $T^*(s) = a/(s+a)$ where $a = \mu \cdot (1-\rho)$, μ is the service rate and ρ is the utilization. Recall that input load from several services is going to the same resource class. Thus, for the combination of resource classes used by service s_i, we have:

$$T^* (s) = \prod_{j \in RC_i^S} \frac{a_{i,j}}{s + a_{i,j}} \tag{5}$$

where $a_{i,j} = \mu_{i,j} \cdot (1 - \rho_j)$. Inverting the transform yields the probability density function of response time, which is integrated to find the cumulative probability distribution function (PDF) of response time, $T_i (y)$. Finally:

$$B_i \left(T_i^{DEF}\right) = Pr \left[\tilde{T}_i > T_i^{DEF}\right] = 1 - T_i \left(T_i^{DEF}\right) \tag{6}$$

Additionally, average response time is typically defined in an SLA and may be found from the Laplace transform as follows:

$$\bar{T}_i = - \left. \frac{dT_i^* (s)}{ds} \right|_{s=0} \tag{7}$$

4 A Numerical Example of SLA Design

The purpose of this section is to go through a complete example and verify the extent to which the method proposed can be useful in designing SLAs, i.e., choosing SLO values. Assume the existence of a single service (the index i is dropped) using three resource classes: a Web resource class (RC_{web}), an application server resource class (RC_{as}) and a database resource class (RC_{db}). In the example, the parameters shown in Table 1 are used, typical for current technology [8]. In that table, tuples such as (a,b,c) represent parameter values for the three resource classes (web, application, database); furthermore, each resource is made up of three components: hardware (hw), operating system (os) and application software (as).

Let us now first get a feeling for the variation of some of these measures. Figure 2 shows how the loss component due to response time (ΔX_i^T) indeed varies as response time rises with increased load. Similarly, one can get a feel for the loss component due to availability (ΔX_i^A) from Figure 3. In that figure, availability is made to improve by changing the number of database machines from 2 to 6, while keeping other infrastructure components constant. The loss due to high response time is very low and is thus not shown in the figure. As one can see, cost increases, loss due to unavailability decreases while "cost + loss" reaches a minimum value for 4 machines.

It is now time to consider the main problem of interest in this paper: that of SLA design. If one were to design the SLA in an ad hoc way, one could approach the problem from the infrastructure side and try to minimize cost while maintaining reasonable service availability and response time. The cheapest infrastructure here is (n_{web}, n_{as}, n_{db}, m_{web}, m_{as}, m_{db})=(1,1,1,1,1,1). However, this design cannot handle the applied load (average response time is very high) due

Table 1. Parameters for example

Parameters	Values	Parameters	Values
T^{DEF}	8 seconds	α_j	(1,1,3)

ϕ	$1 per transaction	$c_{j,k}^{Active}$
		($/month)
			hw =(1100, 1100, 4400)
			os=(165, 165, 165)
			as=(61, 30, 660)

γ	14 transactions per second	$c_{j,k}^{Standby}$
		($/month)
			hw =(1000, 1000, 4000)
			os=(150, 150, 150)
			as=(55, 0, 600)

ΔT	1 month	D_j	(0.05, 0.1, 0.2) seconds
A_j^R (resource availability for R_j)	99.81% (this value is calculated from appropriate MTBF and MTTR values)		

to saturation of the application server. A second try yields (1,2,1,1,2,1) – more power in the application tier. This yields a monthly cost of $9141, and SLOs of (average response time=1.5 s, service availability=95.32%). Since this availability is not typically considered adequate, the designer may increase the number of machines in other tiers yielding a design with infrastructure (3,3,3,1,2,1), cost $22201 and SLOs of (1.5 s, 99.96%). There the designer may rest. We will shortly show that this is not an optimal design.

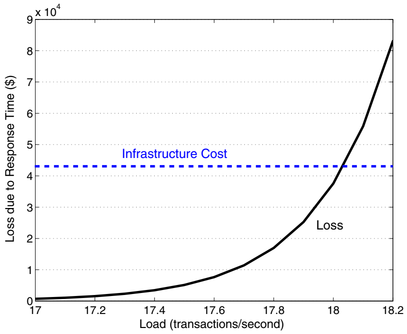


Fig. 2. Effect of Load on Loss

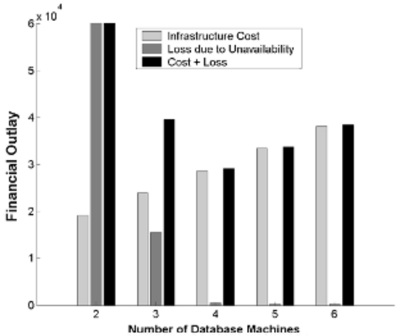


Fig. 3. Sensitivity of Loss due to Redundancy

Alternatively, the designer may base the design on the customer and overdesign with (5,5,5,2,3,1), cost $37152 and SLOs (0.39 s, 99.998%). None of the above design decisions take loss into account. It is instructive to discover the values for loss for the above designs as well as for the design which minimizes the sum of cost plus loss as shown in section 3.4 (see Table 2).

For the best design, the SLOs are (average response time=0.625 s, availability=99.998%). It has lowest overall financial outlay, and the table clearly shows

Table 2. Comparing designs

Infrastructure	**Cost** **($)**	**Loss due to Response ($)**	**Loss due to unavailability ($)**	**Cost plus loss ($)**	**The cost of choosing wrong ($)**
(1,2,1,1,2,1)	9141	20886	1697369	1727396	1698274
(3,3,3,1,2,1)	22201	21902	15428	59531	30409
(5,5,5,2,3,2)	37152	0	608	37760	8638
(3,4,4,1,2,2)**(best)**	28576	0	546	29122	0

the high cost of choosing SLOs in an ad hoc fashion: a wrong choice can cost tens or even hundreds of thousands of dollars per month.

As a final experiment, it is instructive to see that the best design depends quite heavily on the importance of the business process being serviced. If one lessens the importance of the BP by diminishing the average revenue per transaction by a factor of 10, the best design is (2,4,2,1,2,1), cost $17396, total loss $3243 and SLOs: (average response time=1.5 s, availability=99.97%). In this case, a much lower availability is best and the design is cheaper by $11180 a month than if BP importance were not considered.

5 Related Work

Business Impact Management is a very new area of interest to researchers and practitioners that has not yet been consolidated. In the recent past, some problems typically faced in IT management are being studied through a business perspective [1,2,3,4,5,6,7]. Some examples include incident prioritization [2], management of Web Services [5], Business Process Management [4], etc. These references confirm a general tendency to view BIM as a promising way of better linking IT with business objectives. However, these references offer little in terms of formal business impact models to tie the IT layer to BP or business layers. This is one of our main contributions.

Although this paper stresses aspects of SLA Design, it is also licit to view the work as a method for IT infrastructure design (capacity planning). In this particular area, [8] describes a tool – AVED – used for capacity planning to meet performance and availability requirements and [9] describes a methodology for finding minimum-cost designs given a set of requirements. However, none of these references consider the problem of capacity planning from a business perspective, using business metrics. Furthermore, response time considerations are not directly taken into account. Finally, [10] considers the dynamic optimization of infrastructure parameters (such as traffic priorities) with the view of optimizing high-level business objectives such as revenue. It is similar in spirit to the work reported here, although the details are quite different and so are the problems being solved (SLA design is not the problem being considered). The model is solved by simulation whereas our work is analytical.

In the area of SLA design, HP's Open Analytics [11] is a response to the downside of designing SLAs with current practices leading to a more formal approach as presented here. Open Analytics dictates that all assumptions leading to a performance decision must be made explicit and that all technical and financial consequences must be explained. "Open auditable mathematics, rather than wet finger in the air responses to requests [...]" must be used although details are not given.

Management by Contract [12] investigates how IT management can decide when it is better to violate an SLA or to keep compliance, according to a utility function that calculates the business impact of both alternatives. It is similar in spirit to our work, although it does not specifically address the problem of SLA design.

6 Conclusions

This paper has proposed a method whereby best values for Service Level Objectives of an SLA can be chosen through a business perspective. Business considerations are brought into the model by including the business losses sustained when IT components fail or performance is degraded. This is done through an impact model, fully developed in the paper. A numerical example consisting of a single e-commerce business process using a single IT service dependent on three infrastructure tiers (web tier, application tier, database tier) was used to show that the best choice of SLOs can be vastly superior to ad hoc design. A further conclusion is that infrastructure design and the resulting SLOs can be quite dependent on the "importance" of the BPs being serviced: higher-revenue BPs deserve better infrastructure and the method presented shows exactly how much better the infrastructure should be.

Much work can be undertaken to improve the results, among which the following are worth noting: a better availability model (such as presented in [8]) can be used to approximate reality more faithfully; the load applied to the business process can be better modeled by following the Customer Behavior Model Graph approach [13]; variations in the load applied to the BPs should be investigated; more complete impact models should be developed to be able to deal with any kind of BP, not only e-business BPs heavily dependent on IT; finally, the work should be extended to adaptive infrastructures and dynamic provisioning.

Acknowledgments. We would like to acknowledge and thank the Bottom Line Project team. This work was developed in collaboration with HP Brazil R&D.

References

1. V. Machiraju, C. Bartolini and F. Casati (2004), "Technologies for Business Driven IT Management", In L. Cavedon, Z. Maamar, D. Martin, and B. Benatallah (editors) *Extending Web Services Technologies: the Use of Multi-Agent Approaches*, Kluwer Academic Publishers, 2004.

2. C. Bartolini and M. Sallé (2004), “Business Driven Prioritization of Service Incidents”, In Proc. *15th IFIP/IEEE Distributed Systems: Operations and Management (DSOM 2004)*, 15-17 November 2004, Davis, CA, USA.
3. P. Mason, A New Culture for Service-Level Management: Business Impact Management, IDC White Paper.
4. F. Casati, M. Castellanos, U. Dayal, M. Hao, M. Sayal and M.C. Shan, Business Operation Intelligence Research at HP Labs, In *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2002.
5. F. Casati, E. Shan, U. Dayal and M.C. Shan, Business-Oriented Management of Web Services, In *Communications of the ACM*, October 2003.
6. Z. Liu, M. Squillante and J. Wolf, On Maximizing Service-Level Agreement Profits, In *ACM Electronic Commerce Conference*, October 2001.
7. Y. Diao and J. Hellerstein and S. Parekh, A Business-Oriented Approach to the Design of Feedback Loops for Performance Management, In *Proc. of the 12th International Workshop on Distributed Systems: Operations and Management*, 2001.
8. G. Janakiraman, J. Santos, Y. Turner; Automated Multi-Tier System Design for Service Availability, In *Proceedings of the First Workshop on Design of Self-Managing Systems*, June 2003.
9. D. Ardagna, C. Francalanci, A Cost-Oriented Methodology for the Design of Web-Based IT Architectures, In *Proceedings of the 2002 ACM symposium on Applied Computing*, 2004.
10. S. Aiber, D. Gilat, A. Landau, N. Razinkov, A. Sela, and S. Wasserkrug, “Autonomic Self-Optimization According to Business Objectives”, In *Proceedings of the International Conference on Autonomic Computing*, 2004.
11. R. Taylor, C. Tofts; Death by a thousand SLAs: a short study of commercial suicide pacts, HP Technical Report, January 2005.
12. M. Sallé and C. Bartolini (2004), “Management by Contract”, In *Proceedings of the 2004 IEEE/IFIP Network Operations and Management Symposium*, Seoul, Korea, April 2004.
13. D. Menascé, V. Almeida and L. Dowdy, “Performance by Design”, Prentice Hall PTR, 2004.
14. L. Kleinrock, *Queuing Systems, Vol I: Theory*, Wiley, New York, 1975.
15. K. S. Trivedi, *Probability & Statistics with Reliability, Queuing and Computer Science Applications*, Prentice-Hall, 1982.

Generic Policy Conflict Handling Using a priori Models

Bernhard Kempter[1] and Vitalian A. Danciu[2]

[1] Siemens Corporate Technology,
bernhard.kempter@siemens.com
[2] Munich Network Management Team*, University of Munich
danciu@mnm-team.org

Abstract. The promise of policy-based management is lessened by the risk of conflicts between policies. Even with careful conception of the policies it is difficult if not impossible to avoid conflicts completely. However, it is in principle possible to detect and resolve conflicts either statically or at runtime. Taking advantage of existing managed systems models it is even possible to detect and resolve policy conflicts not addressed until now. In this paper we present a generic approach to automated policy conflict detection based on existing knowledge about a managed system. We describe a methodology to derive conflict definitions from invariants of managed systems models, and show how these can be used to detect and resolve policy conflicts automatically.

1 Introduction

As organisations grow – be they corporations, educational facilities or governmental agencies – the number of decision-makers within increases. Business goals formulated by different decision-makers are divergent or conflicting in some cases. When these goals are projected onto IT management, these conflicts will manifest as management conflicts. In principle they will result in conflicting actions, independently of the management architecture deployed, or the associated programming paradigm. Conflicting actions lead to unpredictable results: mild effects could be the failure of single tasks, while more serious cases could lead to faultily configured or malfunctioning systems. Since often business goals are implemented by scripting for management tools, the resolution of conflicts is a task performed after the detection of a conflict and needs to be executed by personnel with insight into the management system, thus incurring high cost. As management evolves in the direction of self-managed systems, automation of conflict handling becomes indispensable.

An important approach to pursuing management goals is the derivation of policy from these goals. Policies at an operational, technical level can be enforced by means of a policy architecture that guides the execution of management actions on a distributed system. Policy-based managment is the only management paradigm that allows, plausibly, conflict detection and resolution. In this paper we present a solution to conflict handling in policy-based systems that exploits a priori models of managed systems.

* The authors wish to thank the members of the Munich Network Management 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, the Munich University of Technology, the University of the Federal Armed Forces and the Leibniz Supercomputing Center.

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 84–96, 2005.

A generic approach to automated resolution of conflicts between obligation policies is still missing, since such conflicts cannot be resolved from the information given in the policies alone. They are dependent on the structure and setup of the managed system. Thus, a model of the managed system is necessary for determining whether a set of policies is in conflict or not.

Driven by the ever increasing complexity of today's systems, organisations create models of their systems. The advent of service-orientation entices them to model systems in detail and as a whole, in contrast to modeling isolated components. Frameworks like the Common Information Model (CIM) [4] provide a base for such efforts. The resulting models describe the nominal state of the deployed system by not only representing attributes of system components but also relations between them, e.g. functional or structural dependencies. Hence, a finished model can be seen as a specification of the managed system in case.

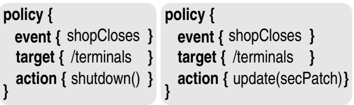
In this paper we demonstrate how such a priori (i.e. existing) models can be leveraged to detect and solve policy conflicts (Section 3). This allows us to address policy conflict types that until now have been inaccessible to automated detection and resolution. The core of the approach is a methodology for deriving reusable, formal conflict definitions from model aspects and management action sets. These definitions yield constraints that allow automated detection of policy conflicts (Section 4). The broad applicability of the methodology is demonstrated by means of a static relationship model for functional dependency. It can also be applied to other static models, such as containment models, as well as dynamic models, e.g. state models. We use the results of the methodology in Section 5, where we show how automated policy conflict detection can be performed and discuss strategies for automated conflict resolution. We present related work regarding policy conflict resolution and selected modeling techniques and standards in Section 6.

2 Models of Managed Systems

In this section we discuss aspects of object oriented models instrumental to automated conflict detection and resolution. We bootstrap the approach to policy conflict handling by assessing a general work process of an administrator who enforces policy by hand. Based on that motivation we discuss characteristics of model hierarchies that are useful to our conflict handling approach.

2.1 Manual Conflict Handling

Consider the two obviously conflicting policies shown in Fig. 1: one policy specifies that all terminals be shut down after working hours, the other specifies that security patches should be installed at that time. A human administrator is able to solve the conflict by allowing the patches to be installed before shutting down the terminals. He assigns an explicit ordering to the policy set to be enforced; in consequence, both policies are enforced and a desired result is achieved.



```
policy {
  event { shopCloses }
  target { /terminals }
  action { shutdown() }
}
```

```
policy {
  event { shopCloses }
  target { /terminals }
  action { update(secPatch)}
}
```

Fig. 1. Simple conflict

In order to find this solution, the administrator uses his knowledge about the managed system: he knows beforehand that patches cannot be installed after the terminals have been shut down. By means of this *a-priori model of the system* he can conclude that he encountered a policy conflict. He uses this model to find an alternative execution path, thus resolving the conflict.

2.2 Hierarchy of Models

To describe systems, we usually model them at some level of abstraction. The models normally cover specific static or dynamic aspects of the system they represent, e.g. states and transitions in that system, its structure or its attributes. Hence, models constitute views from different perspectives onto the system.

A given managed object (MO) is embedded in different kinds of models, e.g. it can be a node of a containment tree and, at the same time, a partition of an automaton describing states of the system.

Fig. 2 gives an example of the abstraction hierarchy of models belonging to or derived from CIM. Traversing the diagram from its top downwards, the level of abstraction decreases and the size of models increases. In common practice, a standard like CIM defines abstract classes and associations that are independent of any managed system or vendors (CIM core schema). The core schema is specialized into customized models (often including some levels of abstraction in the customization as well) to fulfill the requirements of an organisation. The resulting customized model is then instantiated according to the infrastructure it represents.

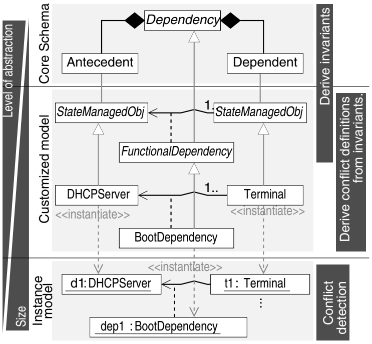


Fig. 2. Hierarchy of models

The diagram shows an example including an abstract dependency from the core schema, the specialization to the abstract class of a functional dependency, and finally to a boot dependency in the customized model. The boot dependency describes that a terminal is dependent on a DHCP server at boot time. Instantiation of the classes of the customized model produces the instance model which represents the managed system at runtime.

Leveraging Model Derivation. The models on adjacent levels are related to each other in that the more concrete model is derived from the more abstract, shown with grayed lines in the diagram. Two different derivation alternatives are employed: inheritance and instantiation, both imparting features to the more concrete model.

This circumstance can be exploited to minimize the effort needed in conflict detection, since any aspect of an abstract model will be present in the more concrete ones. Rightmost in Fig. 2 the activities described in the following sections of this paper are mapped to model abstraction levels. Note that derivation of invariants and conflict definitions (done manually) are performed on the abstract levels, where model size is small or moderate. In contrast, automated conflict handling resides on the most concrete model.

3 Using Models to Support Conflict Handling

Setting for policy-based management. Policies that are to be evaluated concurrently (e.g. because they triggered on the same event) are said to compose a *situation*. This situation may contain a conflict or not. The detection scheme presented in this paper determines the existence of a conflict and identifies the conflicting policies based on a frequently encountered situation definition. The last section hints at how to redefine the situation in order to extend or adapt the conflict handling approach.

Informal conflict notion. Before detailing the application of models, we need to differentiate actual conflicts from other misbehavior of a policy system. The following three common conditions must be satisfied for a policy conflict to be possible ([13])

- Conflicts occur between two or more policies.
- Policies are evaluated concurrently.
- Goals of policies cannot jointly be reached.

In this paper's perspective, these conditions can be summarised as:

• *Constraints extracted from* a priori *models must not be violated by policies in the same* situation. Though we rule out ´conflicts´ occurring from the execution of a single policy, the model based approach could be used to detect such faulty policies as well.

To clarify the settings assumed for a policy-based management scenario, Fig. 3 shows three abstraction planes relevant to the approach presented in this paper. The management plane at the top includes the set of policies, where decisions are made and the execution of operations is initiated.

In the model plane, which corresponds to the instance model in Fig. 2, an implementation view is assumed in addition to the model view depicted. Thus, the MOs also imply agents, specifically policy execution points (PEP) able to enforce policies by executing actions on the underlying infrastructure. From the model perspective, the MOs hold a representation of an element while from the implementation perspective they constitute a middleware layer, obscuring the heterogeneity of the infrastructure. To avoid crowding the diagram, these two perspectives are not differentiated between. Finally, the infrastructure plane at the bottom includes all resources (hardware, applications, services etc.) to be managed.

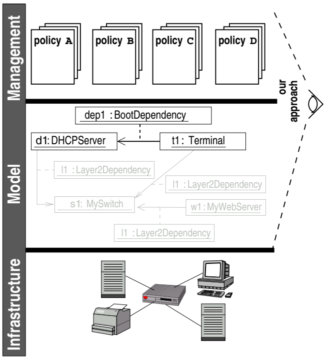


Fig. 3. Model and Views

Required Policy Components. Policies can be specified at different abstraction layers, ranging from corporate or high-level policies at an abstract level, down to operational policies at a technical level. Though the methodology presented in this paper may work at a higher level of abstraction, it is targeted at the operational policy level. It is a requirement that policies be expressed in a formal policy language, rather than in prose.

Policy languages provide different sets of language elements. While the conflict detection methodology is applicable to any language, the latter must provide a minimum expressiveness (see also Fig. 1):

Event. The concept of a *situation* mentioned in Section 3 implies that the language have an event clause. (Examples are time, alarm etc.)

Target. Since our approach focuses on models of the target MOs, we need a target clause stating the MOs (or management domains) which are to be manipulated.

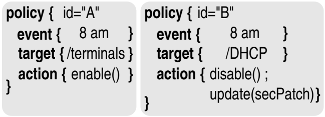
Action. Finally, an action clause is necessary to determine the concrete action to be executed on target objects.

3.1 Approaching Conflict Formalisation

The example in Fig. 4 shows an obvious conflict that cannot be detected or resolved without knowledge from models. It serves as motivation for the approach proposed in this paper while indicating in general the information needed to tackle conflicts of this type. The two policies A and B shown in the figure manipulate DHCP servers resp. the terminals (more precisely: call operations on the managed object boundary of the MOs). The notation in the target field of policies describes a *domain* which is a set of MOs build along management aspects [11].

Using an approach that only considers the management plane from Fig. 3 (i.e. solely the policies themselves) the conflict will not be detected, since the policies address different objects in different domains. Moreover, the goals to enable and disable different MOs are not per se conflicting.

If we consider the management plane and the model plane *in addition* (see Fig. 3), we can determine that the terminal `t1` has a *boot dependency* (which is a kind of a *functional dependency*, see Fig. 2) on the DHCP server `d1`. This circumstance is reflected in the instance model (Fig. 2). To achieve the goal of policy `A` (the dependent terminal can be used), the DHCP server also has be to usable. Policy `B` prohibits this goal by disabling the DHCP server. With the dependency information gathered from the model it becomes obvious that achieving the goals of both policies at the same time is not possible. Thus, a conflict between policies `A` and `B` has been detected.



```
policy { id="A"                  policy { id="B"
  event {   8 am    }              event {      8 am     }
  target { /terminals }            target {    /DHCP    }
  action { enable() }              action { disable() ;
}                                               update(secPatch)}
                                 }
```

Fig. 4. Example of conflicting policies: Is there a conflict between policies `A` and `B`?

3.2 Invariants of Managed Systems

Every managed system is governed by implicit rules resulting from its design. They range from very simple ones (e.g. a unit cannot perform its tasks while switched off) to complex dependencies between components or services. Invariants that formally describe these rules can be extracted from the models of the managed system. Again, different types of models will yield invariants of a different perspective.

A similar concept is found in the integrity constraints common in relational database management systems (RDBMS). An example for their purpose is to ascertain that a data set is not deleted as long as a dependent data set exists. Attempted violations of these constraints are interdicted by the DBMS.

The realization of this concept is eased by the fact that the number of actions is small (for an SQL-DBMS: insert, update, drop ...) as is the number of different data structures (relation, set/row etc). The concise action set found e.g. in DBMS is the result of well-adopted standardization. In systems and service management, the number of available management actions as well as their semantics is far from uniform.

Invariants are Indicators for Conflicts. A management action resulting in the violation of an invariant suggests a management problem, since with it an intrinsic rule of the system has been broken. A conflict between actions is indicated when the combined execution of two or more actions in the same situation results in the breach of an invariant.

4 Conflict Detection: Step by Step to Conflict Definition

In this section we outline the methodology for extracting conflict definitions and detection clauses from models. In the following, *conflict definition* refers to specific kinds or classes of conflicts (as in Fig. 8), not the generic policy conflict as such. A conflict is defined by stating model inherent requirements that are violated when a conflict occurs. Below, we present the steps necessary to distill such conflict definitions based on models at a high abstraction level. These definitions are still valid in the more concrete, derived layers of the model hierarchy, thus reducing the effort for conflict definition.

Step 1: Select a type of model. Any one of the available models can be selected in this step and it makes sense to perform these steps for more than one model. To illustrate the principle by example, we will demonstrate the methodology for functional dependencies.

Example. Selecting functional dependencies as the focused type of association, the level of abstraction to define invariants has to be chosen. To reach optimal reusability of definitions we chose the highest level of abstraction — here it is the level of abstract classes. Fig. 5 shows a section of a object oriented class hierarchy which might be derived from the generic CIM schema.

An MO StateManagedObject has possible states Disabled and Enabled and two methods to change the state. Enabled means that the resource is ready to execute user requests while Disabled means that usage is prohibited. A StateManagedObject can be linked with another StateManagedObject by an association called FunctionalDependency. Thereby, an MO can have the role of a (functional) dependent or an MO provides functionality (antecedent). The exact definition of this association class is made in the next step.

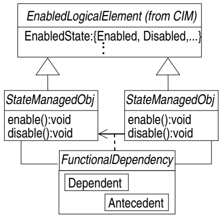


Fig. 5. Detailed customized model of functional dependency

Step 2: Extract invariants from the model. An invariant describes a model aspect in a formal, machine processable way and it can be evaluated to boolean values. Invariants can be classified into general invariants and specific ones. An example for a general invariant of a containment relationship is: the enclosing managed object (MO) must exist at least as long as the contained MOs. Such an invariant is inherent to a model; it is not related to the policies specified for the system. In order to specify invariants, an appropriate language has to be chosen.

Object Constraint Language (OCL). The Unified Modeling Language (UML) defines a formal language to describe constraints for any (UML) model. OCL [12] can be used to define invariants, pre- and postconditions as necessary for our methodology. For our purpose, all invariants are defined for classes and must hold for all instances of that class. Though any equivalent formalisms can be used instead, UML does provide a common language for graphical presentation of object models and OCL offers the opportunity of using an OCL compiler to translate invariants (and consequently conflict definitions) to executable code, thus enabling their direct use.

```
context FunctionalDependency
inv     funcdep:
        self.Dependent → exists (mo | mo.status='Enabled')
        implies self.Antecedent.status='Enabled'
```

Fig. 6. Invariant 1

Example. An invariant consists of two parts: the context part which describes the starting point of the invariant (here it is the class FunctionalDependency from Fig. 5) and the inv part which contains the constraint. To specify an invariant for functional dependencies, we have to reflect which condition has to hold (is always evaluated to ´true´) for the whole life time of that association: the invariant in Fig. 6 states that if an instance exists in the dependent role and its status is Enabled then the antecedent has also to be in its Enabled state to ensure proper execution of the dependent. (The authors are aware that this is only one possible definition out of a huge set. As the paper focus on conflicts we leave a discussion of optimal dependency definition.)

The keyword self refers to an instance of the class FunctionalDependency. With help of a dot you can navigate through the model: self.Dependent is the set of instances of the class `StateManagedObject` which hold the dependent role in this concrete instance (`self`) of `FunctionalDependency`.

Step 3: Derive relationships of invariants to policies. In this step, invariants are mapped to policy actions. For this purpose, the general invariants mentioned in the previous step are considered along with the policies.

While policies contain actions to be executed, invariants do not. Therefore, the effect of the actions on the model needs to be specified. As shown in the example in step 2, all actions changing the state of the associated MOs are described there by describing the effect of an MO´s method as postconditions.

Postcondition 1

```
context StateManagedObject :: disable ( ) : void
   post: self.status = 'Disabled'
```

Postcondition 2

```
context StateManagedObject :: enable ( ) : void
   post: self.status = 'Enabled'
```

Fig. 7. Postcondition1and2

Example. The abstract class `StateManagedObject` consists of two methods `disable()` and `enable()` which are described in OCL (Fig. 7). Postcondition 1 states, that after termination of the method the attribute of the class `StateManagedObject` has the value `Disabled`. Postcondition 2 is defined in the same way.

Step 4: Create conflict definition. The conflict definition "parts" determined in the preceding steps are combined in this step. A conflict definition describes the circumstances in which conflict occurence is certain. Having defined an invariant (step 2) and post conditions for the methods (step 3) this step analyzes if the invariant can be evaluated to ´false´ and if so, a conflict definition is generated.

Example. As the functional dependence has two different ends (dependent and antecedent) and the objects associated have two different methods (`disable()` and `enable()`) there are 4 pairs of methods call to examine (Fig. 8).

For the first pair, the invariant cannot be evaluated to ´false´ in any execution order, hence this pair is always conflict free. The next two pairs are in conflict depending on the execution order (and are therefore race conditions). Conflict definition 1 and 2 in Fig. 8 reflects this situation. The last pair is in conflict disregarding the execution order (see Conflict 3).

To describe a conflict definition OCL is extended by the following keywords: **conflict** to denote the name of the conflict, **conflict space** to identify the actions/operations relevant to the conflict and **refers to** to identify the invariant that the conflict references.

Conflict 1

```
   conflict FuncDepDisable
    context FunctionalDependency
   conflict MO[mo1] :: disable( )
     space: MO[mo2] :: disable( )
       pre: self.Antecedent = mo1
            and self.Dependent = mo2
refers to: inv FunctionalDependency
```

Conflict 2

```
   conflict FuncDepEnable
    context FunctionalDependency
   conflict MO[mo1] :: enable( )
     space: MO[mo2] :: enable( )
       pre: self.Antecedent = mo1
            and self.Dependent = mo2
refers to: inv FunctionalDependency
```

Conflict 3

```
   conflict FuncDep
    context FunctionalDependency
   conflict MO[mo1] :: disable( )
     space: MO[mo2] :: enable( )
       pre: self.Antecedent = mo1
            and self.Dependent = mo2
refers to: inv FunctionalDependency
```

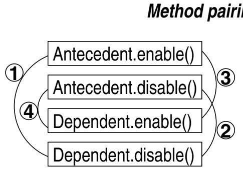


Fig. 8. Conflict definitions and method pairs

As the execution of operations always is the cause conflicts, naming the involved operations is an important part of the definition. The optional precondition narrows the context of the operations when a conflict occurs.

```
Precondition 1
context MO :: disable( ) : void
   pre: self.Antecedent
        implies
          self.Antecedent.Dependent.forall(
          mo | mo.status = 'Disabled')
```

```
Precondition 2
context MO :: enable( ) : void
   pre: self.Antecedent
        implies
          self.Antecedent.Dependent.forall(
          mo | mo.status = 'Enabled')
```

Fig. 9. Preconditions

Step 5: Provide conflict detection clause. For a conflict definition, a detection clause is sought that accounts for the characteristics specific to the conflict definition. In our example, if a `StateManagedObject` is in the role of the antecedent , and wants to `disable()` then it must be ensured that all dependents are already in the state ´Disabled´ (see Precondition 1 in Fig. 9).

With respect to this precondition, a sequential execution of policies could resolve a conflict.

5 Conflict Handling

In this section we show how the concepts developed in Sections 3 and 4 can be applied to detect and handle policy conflicts. During operation of the managed system, policies are triggered by events generated in the system. Before their actions are executed, the set of policies in a situation (see Section 3) are analysed with respect to possible conflicts. If a conflict is detected, the strategies presented in Section 5.2 can be applied to attempt its resolution.

5.1 Application of Conflict Detection

Fig. 12 shows an activity diagram of the algorithm for conflict detection and resolution. Since the algorithm works on sets of policies, it depicts such sets (instead of objects) as input and output of the activities.

To illustrate the conflict handling algorithm, we use the situation shown in Fig. 10 as an example. In addition to the two policies (A and B) from Fig. 4, two other policies are triggered by the same event: one that specifies that the webserver should reread its configuration files (C), and one that enables the printing service (D).

In an example using as few as four policies, some of the steps described in the following may seem redundant. When considering a large number of policies operating on large models, these steps ensure that all situations are handled correctly.

Prerequisites. Three information sets are needed for conflict handling: policies in a situation, e.g. those shown in Fig. 10; conflict definitions, e.g. those described in Section

```
policy { id="A"
  event {   8 am   }
  target { /terminals }
  action { enable() }
}
```

```
policy { id="B"
  event {     8 am    }
  target {    /DHCP   }
  action { disable() ;
           update(secPatch) }
}
```

```
policy { id="C"
  event {   8 am   }
  target { /printer  }
  action { enable() }
}
```

```
policy { id="D"
  event {     8 am    }
  target { /webserver }
  action { rereadConfig() }
}
```

Fig. 10. Example *situation*

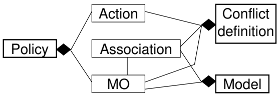


Fig. 11. Information sets

4; models of the system, as shown in the model plane of Fig. 3. These sets are related, as shown in Fig. 11:

The policies contain actions and references to MOs (targets). Actions relevant to policy conflicts are found in the conflict definitions, as are MOs and associations of MOs. In addition, the conflict definition specifies patterns of association between MOs in the models. To perform conflict handling, we leverage the relation between these information sets.

Match Conflict Definitions. The policies in a situation are tested against the conflict definitions acquired by means of the procedure described in Section 4. Specifically, the actions of the policies are compared to the `conflict space` fields of conflict definitions. Matching policies constitute a *matching set* within the potentially conflicting set.

In our example situation, the actions of the policies A - D are matched to the actions of the conflict definitions 1-3 (Fig. 8). It is obvious that the policies containing methods `enable()` and `disable()` will match. Thus, the matching set contains policies A, B and D. These are used as input to the next activity.

Sort by Conflict Definition. From the matching set, a number of sets $cd(i)$ are created, each corresponding to exactly one conflict definition i. The sets may overlap, since a single policy may match several conflict definitions.

In our example, the conflict space of all conflict definitions contain the methods `enable()` and `disable()`. Hence, all the policies in the matching set match all conflict definitions, so that three sets $cd(1)$, $cd(2)$ and $cd(3)$ result, each one of them containing policies A, B, D. Thus: $cd(1) = cd(2) = cd(3) = \{A, B, D\}$

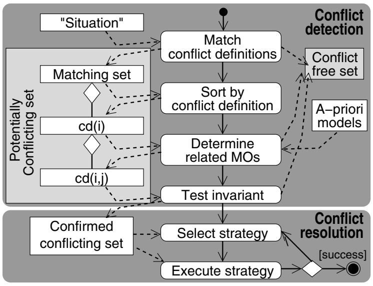


Fig. 12. Conflict handling algorithm

Determine Related MOs. The previous activity has correlated the policy actions and the conflict definitions, thus identifying potentially conflicting policy sets. To violate an invariant, actions must be executed on objects that are related according to the invariant. To test this condition, we compare the targets of every policy in every set $cd(i)$ with MOs referenced in the conflict definitions and determine whether the roles they carry in the model (e.g. a dependency) corresponds. Again, the policies in a set $cd(i)$ can be related to several model partitions, and several roles j may have to be tested. All possible combinations of policies from a set $cd(i)$ result in sets $cd(i, j)$. As in the previous step, the resulting $cd(i, j)$ may overlap.

To clarify this step, consider the following procedure:

1. Select a set to begin with, e.g. $cd(1)$
2. Find all instances j of the class from the invariant of i: results in all instances of `FunctionalDependency`.
3. Determine targets of policies in the set $cd(1)$ selected: result is `terminal`, `DHCPserv` and `printer`
4. For all j, find instances containing the targets above: results in sets $cd(1, j)$, where j is the current instance of i's invariant's class.
5. Repeat the steps for all remaining $cd(i)$.

The example model only has one association that is relevant for conflict detection, thus the only set yielded by this procedure is $cd(3, 1)$: it matches the pattern Antecedent.disable/Dependent.enable. Since there is no dependency between the `printer` MO and the other two MOs, only policies A and B are left in the set.

At this point, both actions and targets have been accounted for. Further analysis of a set $cd(i, j)$ can be performed disregarding the other sets.

Test Invariant. The previous activity has created sets that correspond to single conflict definitions and contain only policies with a high potential of conflict. This activity corresponds to a simulation of the execution of the policy actions. It divides the sets yielded by the previous step into a *confirmed conflicting set* of policies and conflict-free policies. For each policy set $cd(i, j)$, the invariant referenced in the conflict definition i is tested and evaluated under the assumption that the actions of the policies in the set are executed. Since policies are executed in parallel, the invariant must be tested for all permutations of the serialization of the set. If the set passes all tests, the policies in the set are released from the potentially conflicting set into the conflict-free set. If the invariant evaluates to *false* for at least one permutation, the set is conflicting.

In our example, the terminal is dependent on the DHCP server in that that service must be available for the terminal to boot. The invariant of conflict definition 3 is violated, since the object in the dependent role (terminal) is enabled, while the object in the antecedent role (DHCP server) is disabled. In consequence, the set $cd(3, 1)$ is transferred to the confirmed conflicting set.

Select/execute Strategy. For each conflicting $cd(i, j)$, a resolution strategy must be selected and executed. As shown in Fig. 12, more than one of the strategies discussed in Section 5.2 may be tested, resorting to Strategy A if all others failed. An optimal selection of strategy depends on the number of policies in a set, how time-critical their execution might be and possibly other factors not taken into consideration yet. The effectiveness of the resolution strategy can be tested by applying the same steps as for conflict detection to a modified set of policies.

For our example, a sequential execution of the conflicting policies in appropriate order resolves the conflict (let the terminal boot before disabling the DHCP server).

5.2 Strategies for Conflict Resolution

As discussed in Section 5.1, conflict detection determines a policy set where two or more policies are in conflict with each other. Once this set is known, attempts can be made to resolve the conflicts in an automated fashion, or at least minimize their impact. This section presents strategies to that end.

Existing strategies exhibit an all-out approach to conflict resolution, as the following two alternatives show:

A. The most drastic measure is for the policy service to abstain from enforcing *any* of the policies. This could prove to be a viable approach in applications that are not time critical, but it still requires manual intervention.

B. Another approach found in the literature ([5,8]) is to enforce only the policy with the highest priority in the set – assuming policies have been assigned priorities. This scheme is often mentioned in the context of quality of service policy, and its usefulness may be constrained to that niche.

Strategies using the automated conflict detection presented in this paper allow for a differentiated resolution. Since conflicts can be detected in an automated manner in any set of policies, combinations of the policies in the set can be tried. This yields the following strategies:

C. Try to enforce as many policies as possible, excluding the minimal number of policies for the set to be conflict free. We can determine the set to be enforced by iteratively applying conflict detection to parts of the conflict set.

D. Serialising parts of the policy set and finding an appropriate synchronous enforcement order can resolve the conflict in some cases. This solution appears to be the least invasive, though it will slow down the enforcement of the policies. Again, the enforcement order is found by applying the conflict detection algorithm to permutations of the conflict set.

E. Create conflict free subsets and serialise the enforcement of subsets. This is an optimisation of the above strategy. While the policy subsets are enforced synchronously, the enforcement of the policies in one subset can be parallelised. Normally, i.e. in cases where the number of sets is small compared to the mean number of policies per set, this strategy should execute faster.

These alternatives are orthogonal: any combination of serialisation and reduction of the policy set is possible. Unfortunately, seeking the optimal solution implies testing a large number of policy sets, which may not be practical to do at runtime.

6 Related Work

Policy conflict handling. We imposed the requirement that the conflict handling scheme presented in this paper be independent of a specific policy language, the type of policy and overlap of policy domains as a necessary prerequisite. In the following, we present work related to the approach presented here.

At Imperial College much valuable work in the area of policy conflicts has been published. The result of [10] is a conflict classification. Conflicts are classified along the number of overlaps (at least one) of domains (given in subject, target or action) between two or more policies.

Another conflict detection approach exploiting domain overlapping is found in [8]. It focuses on modality conflicts, where policies are typed (positive and negative authorisation and obligation policy). A triple overlap (subject and target and action) indicates a conflict. This approach is effective in detecting conflicts between authorisation and obligation policies, but limited regarding conflicts between obligation policies.

Damianou [5] uses so called meta policies, which are part of the Ponder specification, to formalize policy conflicts. A meta policy specifies constraints regarding a set of policies. As this approach is language specific and not all policy languages support the concept of constraint-based meta policies, the general applicability is limited. Also, to apply the general-purpose tool of meta policies to conflict handling, a methodology for the specification of appropriate meta policies would have to be created.

The scope of [1] is to support policy refinement. A formal language (event calculus) is used to represent the state of a system allow reasoning about possible future states. Policy language, policy execution and the managed system are formalised using event calculus. Based on the resulting model, conflicts can be defined in event calculus, overlap of domains being a prerequisite for conflict definition.

This approach takes into account the managed system allowing calculus representation of static and dynamic aspects of a whole managed system. However, since since special models must be created, the effort introduced seems to be quite high, especially when considering large scale, complex systems.

In [2,3] a policy conflict resolution approach is shown for the Policy Description Language $\mathcal{PDL}$, a rule-based language which omits the policy element subject and target. Conflicts are defined by monitors which evaluate action constraints. An action constraint has the form: `never Action1 ∧...∧ ActionN ∧ condition`. A methodology to derive action constraints is not given, also the application of the approach to other policy languages and architectures is not discussed.

Management Object Modeling. To be able to derive conflict definitions we need the concept of object orientation especially the concept of classes and methods. Management classes provide abstraction of resources for management purposes.

The instances of a management class and their embedding in a management information base (MIB) is standardised. Well known object oriented MIBs are DMTF's Common Information Model (CIM)[4] and ISO's Structure of Management Information (SMI) [6,7]. For these standards our approach can be applied directly.

For IETF's SNMP-SMI [9] (also known as Internet MIB), which is not object oriented, a wrapper needs to be designed in order to allow our approach to be applied to MIB attributes.

7 Conclusions

In this paper we have discussed an approach to conflict handling relying on a priori models. Different types of models represent static and dynamic aspects of managed systems. They can be leveraged to derive invariants that, in turn, yield conflict definitions. Aided by these conflict definitions, policy sets can be checked for conflicts either statically or at runtime. In the following we summarize the key concepts of the paper and point to further topics of study.

We presented a methodology to derive conflict definitions from object oriented management models. In addition, we have shown how to perform automated conflict detection based on these conflict definitions. Also, we presented conflict resolution strategies for the policy sets found to be in conflict. The approach presented is generic in that it has no dependency regarding type of policy, the policy language used or management model type. Merely three policy elements are prerequisite to using the approach.

Yet, several important topics of study in the area of conflict handling remain. An important issue would be the integration of conflict handling methods for different types of policies, e.g. approaches targeting authorisation policies using a model based scheme.

We assumed that a *situation* consists of policies that have been triggered by one event. Keeping in mind that a situation is merely a set of policies, the same concepts can be extended to support policy sets created in other ways, e.g. by observing a sequence of events. For that purpose, the only thing that needs to be changed is the definition of the situation itself. As a related issue, using invariants to identify faulty policy specification seems to be a rewarding topic.

References

1. Arosha K. Bandara, Emil C. Lupu, and Alessandra Russo. Using event calculus to formalise policy specification and analysis. In *Proceedings of HPOVUA 2003*, 2003.
2. J. Chomicki, J. Lobo, and S. Naqvi. A logic programming approach to conflict resolution in policy management. In *7th International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, pages 121–132, Breckenridge, Colorado, Morgan Kaufman, 2000.
3. J. Chomicki, J. Lobo, and S. Naqvi. Conflict resolution using logic programming. *Transaction on Knowledge and Data Engineering*, 15(1):244–249, 2003.
4. Common Information Model (CIM) Specification Version 2.8. Specification, January 2004.
5. N. C. Damianou. *A Policy Framework for Management of Distributed Systems*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, Department of Computing, February 2002.
6. Information Technology – Open Systems Interconnection – Structure of Management Information – Part 4: Guidelines for the Definition of Managed Objects. IS 10165-4, International Organization for Standardization and International Electrotechnical Committee, 1992.
7. Information Technology – Open Systems Interconnection – Structure of Management Information – Part 7: General Relationship Model. IS 10165-7, International Organization for Standardization and International Electrotechnical Committee, 1997.
8. Emil C. Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering*, 25(6):852–869, November 1999.
9. K. McCloghrie and M.T. Rose. RFC 1065: Structure and identification of management information fo r tcp/ip-based internets. RFC, Internet Engineering Task Force (IETF), August 1988.
10. Jonathan D. Moffett and Morris S. Sloman. Policy conflict analysis in distributed system management. *Journal of Organizational Computing*, 1993.
11. Morris S. Sloman and Kevin Twidle. *Domains: A Framework for Structuring Management Policy*, chapter 16. 1994.
12. OMG Unified Modeling Language Specification, Version 1.5. Technical Report formal/03-03-01, Object Management Group, March 2003. http://www.omg.org/cgi-bin/doc?formal/03-03-01.
13. A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser. RFC 3198: Terminology for policy-based management. RFC, Internet Engineering Task Force (IETF), November 2001.

An Approach to Understanding Policy Based on Autonomy and Voluntary Cooperation

Mark Burgess

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

Abstract. Presently, there is no satisfactory model for dealing with political autonomy of agents in policy based management. A theory of atomic policy units called 'promises' is therefore discussed. Using promises, a global authority is not required to build conventional management abstractions, but work is needed to bind peers into a traditional authoritative structure. The construction of promises is precise, if tedious, but can be simplified graphically to reason about the distributed effect of autonomous policy. Immediate applications include resolving the problem of policy conflicts in autonomous networks.

1 Introduction

One of the problems in discussing policy based management of distributed systems[1,2] is the assumption that all of the nodes in a network will follow a consistent set of rules. For this to be true, we need either an external authority to impose a consistent policy from a bird's eye view, or a number of independent agents to collaborate in a way that settles on a 'consistent' picture autonomously.

Political autonomy is the key problem that one has to deal with in ad hoc / peer-to-peer networks, and in pervasive computing. When the power to decide policy is delegated to individuals, orders cannot be issued from a governing entity: consistency and concensus must arise purely by *voluntary cooperation*. There is no current model for discussing systems in this situation.

This paper outlines a theory for the latter, and in doing so provides a way to achieve the former. The details of this theory require a far more extensive and technical discussion than may be presented in this short contribution; details must follow elsewhere.

It has been clear to many authors that the way to secure a clear and consistent picture of policy, in complex environments, is through the use of formal methods. But what formalism do we have to express the necessary issues? Previous attempts to discuss the consistency of distributed policy have achieved varying degrees of success, but have ultimately fallen short of being useful tools except in rather limited arenas. For example:

- Modal logics: these require one to formulate hypotheses that can be checked as true/false propositions. This is not the way system administrators work.

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 97–108, 2005.

– The π-calculus: has attractive features but focuses on issues that are too low-level for management. It describes systems in terms of states and transitions rather than policies (constraints about states)[3].
– Implementations like IPSec[4,5], Ponder[6] etc. these do not take explicitly into account the autonomy of agents and thus while these implement policies well enough, they are difficult to submit to analysis.

In each of the latter examples, one tends to fight two separate battles: the battle for an optimal mode of expression and the battle for an intuitive interface to an existing system. For example, consider a set of files and directories, which we want to have certain permissions. One has a notion of policy as a specification of the permission attributes of these files. Policy suggests that we should group items by their attributes. The existing system has its own idea of grouping structures: directories. A simple example of this is the following:

```
ACL1:                               ACL2:
 1. READ-WRITE /directory            1. READ-ONLY  /directory/file
 2. READ-ONLY  /directory/file       2. READ-WRITE /directory
```

Without clear semantics (e.g. first rule wins) there is now an ordering ambiguity. The two rules overlap in the specifically named “file”, because we have used a description based on overriding the collection of objects implicitly in “directory”.

In a real system, a directory grouping is the simplest way to refer to this collection of objects. However, this is not the correct classification of the attributes: there is a conflict of interest. How can we solve this kind of problem?

In the theory of system maintenance[7], one builds up consistent and stable structures by imposing independent, atomic operations, satisfying certain constraints[8,9]. By making the building blocks primitive and having special properties, we ensure consistency. One would like a similar construction for all kinds of policy in human-computer management, so that stable relationships between different activities can be constructed without excessive ambiguity or analytical effort. This paper justifies such a formalism in a form that can be approached through a number of simplifications. It can be applied to network or host configuration, and it is proposed as a unifying paradigm for autonomous management with cfengine[10].

2 Policy with Autonomy

By a policy we mean the ability to assert arbitrary constraints of the behaviour of objects and agents in a system. The most general kind of system one can construct is a collection of objects, each with its own attributes, and each with its own policy. A policy can also be quite general: e.g. policy about behaviour, policy about configuration, or policy about interactions with others.

In a network of *autonomous* systems, an agent is only concerned with assertions about its own policy; no external agent can tell it what to do, without its consent. This is the crucial difference between autonomy and centralized management, and it will be the starting point here (imagine privately owned devices wandering around a shopping mall).

Requirement 1 (Autonomy). *No agent can force any other agent to accept or transmit information, alter its state, or otherwise change its behaviour.*

(An attempt by one agent to change the state of another might be regarded as a definition of an attack.) This scenario is both easier and harder to analyze than the conventional assumption of a system wide policy. It is easier, because it removes many possible causes of conflict and inconsistency. It is harder because one must then put back all of that complexity, by hand, to show how such individual agents can form collaborative structures, free of conflict.

The strategy in this paper is to decompose a system into its autonomous pieces and to describe the interactions fully, so that inconsistencies become explicit. In this way, we discover the emergent policy in the swarm of autonomous agents.

3 Promises

The analysis of 'promises' is naturally motivated by the theory of games and voluntary cooperation[11,12] and has, to the author's knowledge, only previously been mentioned in a recent context of economics[13].

A promise is a general and abstract unit of intent. Promises, between agents, can deal with things like quality of service, quality of behaviour, specifications of state, etc. Policies of various types have been identified. For instance, in the Ponder model[6], one has authorizations (promises to grant access) and obligations (promises to follow up on a different promise) or dependency, etc. These can all be translated into the notion of promises.

Consider, then, a set of autonomous agents of objects represented as nodes $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ in a graph.

Definition 1 (Promise). *A promise is a labelled directed edge (link) that connects two nodes. The promise label represents a specifically intended range of behaviour χ from within a domain of possible behaviours. i.e. $n_1 \xrightarrow{\chi} n_2$. A promise is thus made by a node n_1 to a node n_2. A promise is assumed to be always kept.*

Although it will be important, at a later stage, to discuss whether or not promises are kept, we wish to avoid this issue in the initial discussion; we assume it to be true. Notice also that, in the definition, the agent-nodes, between which promises are made, are kept separate from the constraints between them. This is important for avoiding the kinds of ordering ambiguities alluded to in the introduction.

Example 1 (Service Level Agreement (SLA)). Agent n_1 promises agent n_2 to provide service of type 'database access in time q', τ is the type domain $q \in [0, \infty]$ and the constraint $\chi(q) : 0 < q < 10\text{ms}$.

The formulation of a promise, above, has obvious characteristics of a directed graph. It is not a particularly novel construction. It bears a passing resemblance to the theory of capabilities in ref. [14], for instance. Graphs have many desirable properties for defining relationships between entities[15], and there is good reason to retain these properties in describing the relationships between agents. In subsequent work, it will become clear that graphs will prove a useful abstraction of themselves, for management; it is possible to transform graphs and use their spectral properties to discover useful and important properties for management[16,17,18].

Two special types of promise will be identified below, in order to rebuild conventional structures from these basic atoms.

- A promise to agree to behave like another.
- A promise to utilize the promise of another.

The first of these is essential for defining groups, roles and social structures with concensus behaviour. The latter is crucial for client-server interactions, dependencies and access control. The rest of this paper is about logically combining individual promises into collective and consistent policies that allow cooperation between autonomous agents.

4 What Is an Inconsistency?

In the extreme case, in which every agent were independent and could only see its own world, there would be no need to speak of inconsistency: unless agents have agreed to be similar, they can do as they please. The only problem that might occur is if an agent promised two contradictory things to a second agent.

Definition 2 (Broken promise). *A promise of χ_1 from agent n_1 to agent n_2 is said to be broken if there exists another promise from n_1 to n_2, of χ_2, in which $\chi_1 \neq \chi_2$.*

This definition is very simple, and becomes most powerful when one identifies promise *types* which is beyond the present scope. It says that an agent can only break its own promises: if an agent promises two different things, it has broken both of its promises. One might feel the need to define ‘redundant’ promises as being different from broken promises, e.g. if one promise merely extends another then the other is unnecessary; but this opens up an unnecessary subjectivity into the comparison and leads us into trouble straight away. The definition unambiguously identifies a conflict of intention and it can be left up to a human to decide which of the promises is correct, incorrect, redundant etc.

5 Promise Analysis

Logic is a way of analysing the consistency of assumptions. It is based on the truth or falsity of collections of propositions $p_1, p_2, \ldots$. One must formulate these propositions in advance and then use a set of assumptions to determine their status. The advantage of logic is that is admits the concept of a proof.

Is there a logic that is suitable for analyzing promises? Modal logic has been considered as one possibility, and some authors have made progress in using modal logics in restricted models[19,20]. However, there are basic problems with modal logics that limit their usefulness[21].

More pragmatically, logic alone does not usually get us far in engineering. We do not usually want to say things like "it is true that $1 + 1 = 2$"? Rather we want a system, giving true answers, which allows us to compute the value of $1 + 1$, because we do not know it in advance. Ultimately we would like such a calculational framework for combining the effects of multiple promises. Nevertheless, let us set aside such practical considerations for now, and consider the limitations of modal logical formalism in the presence of autonomy.

5.1 Modal Logic and Kripke Semantics

Why have formalisms for finding inconsistent policies proven to be so difficult? A clue to what is going wrong lies in the many worlds interpretation of the modal logics[22]. In the modal logics, one makes propositions p, q etc., which are either true or false, under certain interpretations. One then introduces modal operators that ascribe certain properties to those propositions, and one seeks a consistent language of such strings.

Modal operators are written in a variety of notations, most often with $\Box$ or $\diamond$. Thus one can say $\Box\, p$, meaning "it is necessary that p be true", and variations on this theme:

$\Box\, p$	$\diamond p = \neg\Box\,\neg p$
It is necessary that p	It is possible that p
It is obligatory that p	It is allowed that p
It is always true that p	It sometimes true that p

A system in which one classifies propositions into "obligatory", "allowed" and "forbidden" could easily seem to be a way to codify policy, and this notion has been explored[19,20,23,21].

Well known difficulties in interpreting modal logics are dealt with using Kripke semantics[24]. Kripke introduced a 'validity function' $v(p, w) \in \{T, F\}$, in which a proposition p is classified as being either true of false in a specific 'world' w. Worlds are usually collections of observers or agents in a system.

Consider the formulation of a logic of promises, starting with the idea of a 'promise' operator.

- $\Box\, p =$ it is promised that p be true.
- $\diamond p = \neg\Box\,\neg p =$ it is unspecified whether p is true.
- $\Box\,\neg p =$ it is promised that p will not be true.

and a validity function $v(\cdot, \cdot)$.

5.2 Single Promises

A promise is something that is shared between a sender and a recipient. It is not a property of agents, as in usual modal logics, but of a pair of agents. Logic says nothing about this topology of a promise (indeed, we would like to keep this separate, for reasons that become clearer in section 5.7), so one attempts to build this into the semantics.

Consider the example of the Service Level Agreement, above, and let p mean "Will provide data in less than 10ms". How shall we express the idea that a node n_1 promises a node n_2 this proposition? Consider the following statement:

$$\Box\, p, \quad v(p, n_1) = T. \tag{1}$$

This means that it is true that p is promised at node n_1, i.e. node 1 promises to provide data in less than 10ms – but to whom? Clearly, we must also provide a recipient. Suppose, we try to include the recipient in the same world as the sender? i.e.

$$\Box\, p, \quad v(p, \{n_1, n_2\}) = T. \tag{2}$$

However, this means that both nodes n_1 and n_2 promise to deliver data in less than 10ms. This is not what we need; a recipient is still unspecified. Clearly what we want is to define promises on a different set of worlds: the set of possible links or *edges* between nodes. There are $N(N-1)$ such directed links. Thus, we may write:

$$\Box\, p, \quad v(p, n_1 \to n_2) = T. \tag{3}$$

This is now a unique one-way assertion about a promise from one agent to another. A promise becomes a tuple $\langle \tau, p, \ell \rangle$, where τ is a theme or promise-type (e.g. Web service), p is a proposition (e.g.deliver data in less than 10ms) about how behaviour is to be constrained, and ℓ is a link or edge over which the promise is to be kept. All policies can be written this way, by inventing fictitious services. Also, since every autonomous promise will have this form, the modal/semantic content is trivial and a simplified notation could be used.

5.3 Regional or Collective Promises from Kripke Semantics?

Kripke structures suggest ways of defining regions over which promises might be consistently defined, and hence a way of making uniform policies. For example, a way of unifying two agents n_1, n_2 with a common policy, would be for them both to make the same promise to a third party n_3:

$$\Box\, p, \quad v(p, \{n_1 \to n_3, n_2 \to n_3\}) = T. \tag{4}$$

However, there is a fundamental flaw in this thinking. The existence of such a function that unifies links, originating from more than a single agent-node, is contrary to the fundamental assumption of autonomy. There is no authority in

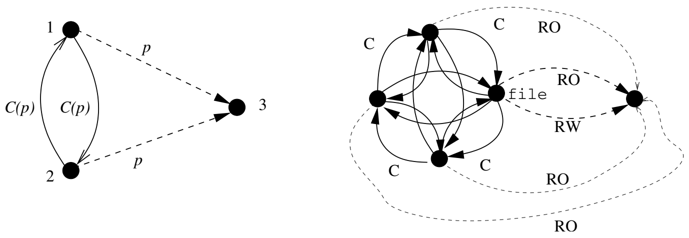


Fig. 1. (Left) Cooperation and the use of third parties to measure the equivalence of agent-nodes in a region. Agents form groups and roles by agreeing to cooperate about policy. (Right) This is how the overlapping file-in-directory rule problem appears in terms of promises to an external agent. An explicit broken promise is asserted by file, in spite of agreements to form a cooperative structure.

this picture that has the ability to assert this uniformity of policy. Thus, while it might occur by fortuitous coincidence that p is true over a collection of links, we are not permitted to *specify* it or demand it. Each source-node has to make up its own mind. The logic verifies, but it is not a tool for understanding construction.

What is required is a rule-based construction that allows independent agents to come together and form structures that span several nodes, by *voluntary cooperation.* Such an agreement has to be made between every pair of nodes involved in the cooperative structure. We summarize this with the following:

Requirement 2 (Cooperative promise rule). *For two agents to guarantee the same promise, one requires a special type of promise: the promise to cooperate with neighbouring agent-nodes, about basic promise themes.*

A complete structure looks like this:

- n_1 promises p to n_3.
- n_2 promises n_1 to collaborate about p (denote this as a promise $C(p)$).
- n_1 promises n_2 to collaborate about p (denote this as a promise $C(p)$).
- n_2 promises p to n_3

By measuring p from both n_1 and n_2, n_3 acts as a judge of their compliance with the mutual agreements between them (see fig. 1). This allows the basis of a theory of measurement, by third party monitors, in collaborative networks. It also shows how to properly define structures in the file-directory example (see fig 1).

5.4 Dependencies and Handshakes

Even networks of autonomous agents have to collaborate and delegate tasks, depending on one another to fulfill promised services. We must find a way of expressing dependency relationships without violating the primary assumption of autonomy.

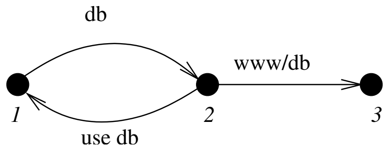


Fig. 2. Turning a conditional dependency into a real promise. The necessary structure is shown in graphical form.

Consider three agents n_1, n_2, n_3, a database server, a web server and a client. We imagine that the client obtains a web service from the web server, which, in turn, gets its data from a database. Define propositions and validities:

- $p_1 =$ "will send database data in less than 5ms", $v(p_1, n_1 \rightarrow n_2) = T$.
- $p_2 =$ "will send web data in less than 10 ms", $v(p_2, n_2 \rightarrow n_3) = T$.

These two promises might, at first, appear to define a collaboration between the two servers to provide a promise of service to the client, but they do not.

The promise to serve data from $n_1 \rightarrow n_2$ is in no way connected to the promise to deliver data from $n_2 \rightarrow n_3$:

- n_2 has no obligation to use the data promised by n_1.
- n_2 promises its web service regardless of what n_1 promises.
- Neither n_1 nor n_3 can force n_2 to act as a conduit for database and client.

We have already established that it would not help to extend the validity function to try to group the three nodes into a Kripke 'world'. Rather, what is needed is a structure that complete the backwards promises to *utilize* promised services – promises that completes a *handshake* between the autonomous agents. We require:

- A promise to uphold p_1 from $n_1 \rightarrow n_2$.
- An acceptance promise, to use the promised data from $n_2 \rightarrow n_1$.
- A conditional promise from $n_2 \rightarrow n_3$ to uphold p_2 iff p_1 is both present and accepted.

Thus, three components are required to make a dependent promise (see fig. 2). This requirement cannot be derived logically; rather, we must specify it as part of the semantics of autonomy.

Requirement 3 (Acceptance/usage promise rule). *Autonomy requires an agent to explicitly accept a promise that has been made, when it will be used to derive a dependent promise.*

One thus identifies a second special type of promise: the "usage" or "acceptance" promise.

5.5 Autonomous, Voluntary Cooperation

What use is this construction? First, it advances the manifesto of making all policy decisions explicit. In the example in fig. 2, it shows explicitly the roles and responsibilities of each of the agents in the diagram. Furthermore, the graphical representation of these promises is quite intuitive and easy to understand. The construction has two implications:

1. The component atoms (promises) are all visible, so the inconsistencies of a larger policy can be determined by the presence or absence of a specific link in the labelled graph of promises, according to the rules.
2. One can provide basic recipes (handshakes etc.) for building concensus and agent "societies", without hiding assumptions. This is important in pervasive computing, where agents truly are politically autonomous and every promise must be explicit.

The one issue that we have not discussed is the question of how cooperative agreements are arrived at. This is a question that has been discussed in the context of cooperative game theory[25,11], and will be elaborated on in a future paper[26]. Once again, it has to do with the human aspect of collaboration. The reader can excerise imagination in introducing fictitious, intermediate agents to deal with issues such as shared memory and resources.

5.6 Causality and Graph Logic

As an addendum to this discussion, consider *temporal logic*: this is a branch of modal logic, in which an agent evolves from one Kripke world into another, according to a causal sequence, which normally represents time. In temporal logic, each new time-step is a new Kripke world, and the truth or falsity of propositions can span sequences of worlds, forming graph-like structures. Although time is not important in *declaring* policy, it is worth asking whether a logic based on a graph of worlds could be used to discuss the collaborative aspects of policy. Indeed, some authors have proposed using temporal logic and derivative formalisms to discuss the behaviour of policy, and modelling the evolution systems in interesting ways[27,28,29].

The basic objection to thinking in these terms is, once again, autonomy. In temporal logic, one must basically know the way in which the propositions will evolve with time, i.e. across the entire ordered graph. That presupposes that such a structure can be written down by an authority for the every world; it supposes the existence of a global evolution operator, or master plan for the agents in a network. No such structure exists, *a priori*. It remains an open question whether causality is relevant to policy specification.

5.7 Interlopers: Transference of Responsibility

One of the difficult problems of policy consistency is in transferring responsibilities from one agent to another: when an agent acts as as a conduit or interloper for another. Consider agents a, b and c, and suppose that b has a resource B which it can promise to others. How might b express to a: "You may have access to B, but do not pass it on to c"?

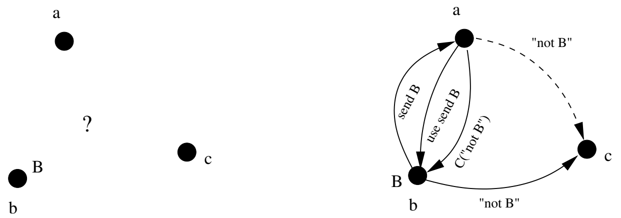


Fig. 3. Transference of responsibility

The difficulty in this promise is that the promise itself refers to a third party, and this mixes link-worlds with constraints. As a single promise, this desire is not implementable in the proposed scheme:

- It refers to B, which a has no access to, or prior knowledge of.
- If refers to a potential promise from a to c, which is unspecified.
- It preempts a promise from a to b to never give B along $a \rightarrow c$.

There is a straightforward resolution that maintains the autonomy of the nodes, the principle of separation between nodes and constraints, and which makes the roles of the three parties explicit. We note that node b cannot order node a to do anything. Rather, the agents must set up an agreement about their wishes. This also reveals that fact that the original promise is vague and inconsistent, in the first instance, since b never promises that it will not give B to c itself. The solution requires a cooperative agreement (see fig. 3).

- First we must give a access to B by setting up the handshake promises: i) from $b \rightarrow a$, "send B", ii) from $a \rightarrow b$, accept/use "send B".
- Then b must make a consistent promise not to send B from $b \rightarrow c$, by promising "not B" along this link.
- Finally, a promises b to cooperate with b's promises about "not B", by promising to cooperate with "not B" along $a \rightarrow b$. This implies the dotted line in the figure that it will obey an equivalent promise "not B" from $a \rightarrow c$, which could also be made explicit.

At first glance, this might seem like a lot of work for express a simple sentence. The benefit of the construction, however, it that is preserves the basic principles of make every promise explicit, and separating agents-nodes from their intentions. This will be crucial to avoiding the contradictions and ambiguities of other schemes.

6 Conclusions

A graphical scheme for analysing autonomous promises has been outlined in a stripped-down form. Cooperative behaviour requires the presence of mutual agreements between nodes. The value of the promise idiom is to make difficult algebraic constraints into a simple graphical technique that is intuitive for management. A number of theorems can be proved about promises (elsewhere). The promise paradigm forces one to confront the fundamental issues in cooperative behaviour, and can be used to build up systems from scratch, seeing the inconsistencies that arise visually. This also opens the way to make analysis tools and incorporate a wider range of policies in cfengine[10].

In such a short paper, it is not possible to expand on the detailed definitions, proofs or numerous applications of this idea. Some applications include, the analysis of management conflicts (especially in autonomous agencies e.g. in BGP), identification of important and vulnerable nodes, by spectral analysis, and providing a language for a general theory of pervasive, autonomous computing. These will be discussed in future work.

Acknowledgement. I am grateful to Jan Bergstra, Siri Fagernes and Lars Kristiansen for helpful discussions.

References

1. M.S. Sloman and J. Moffet. Policy hierarchies for distributed systems management. *Journal of Network and System Management*, 11(9):1404, 1993.
2. E.C. Lupu and M. Sloman. Towards a role based framework for distributed systems management. *Journal of Network and Systems Management*, **5**, 1996.
3. J. Parrow. *An Introduction to the π-Calculus, in The Handbook of Process Algebra*, page 479. Elsevier, Amsterdam, 2001.
4. Z. Fu and S.F. Wu. Automatic generation of ipsec/vpn security policies in an intradomain environment. *Proceedings of the 12th internation workshop on Distributed System Operation and Management (IFIP/IEEE).*, INRIA Press:279, 2001.
5. R. Sailer, A. Acharya, M. Beigi, R. Jennings, and D. Verma. Ipsecvalidate - a tool to validate ipsec configurations. *Proceedings of the Fifteenth Systems Administration Conference (LISA XV) (USENIX Association: Berkeley, CA)*, page 19, 2001.
6. N. Damianou, N. Dulay, E.C. Lupu, and M. Sloman. Ponder: a language for specifying security and management policies for distributed systems. *Imperial College Research Report DoC 2000/1*, 2000.
7. M. Burgess. On the theory of system administration. *Science of Computer Programming*, 49:1, 2003.
8. A. Couch and N. Daniels. The maelstrom: Network service debugging via "ineffective procedures". *Proceedings of the Fifteenth Systems Administration Conference (LISA XV) (USENIX Association: Berkeley, CA)*, page 63, 2001.
9. M. Burgess. Cfengine's immunity model of evolving configuration management. *Science of Computer Programming*, 51:197, 2004.
10. M. Burgess. A site configuration engine. *Computing systems (MIT Press: Cambridge MA)*, 8:309, 1995.
11. R. Axelrod. *The Complexity of Cooperation: Agent-based Models of Competition and Collaboration.* Princeton Studies in Complexity, Princeton, 1997.
12. R. Axelrod. *The Evolution of Co-operation.* Penguin Books, 1990 (1984).
13. J.D. Carrillo and M. Dewatripont. Promises, promises. Technical Report 172782000000000058, UCLA Department of Economics, Levines's Bibliography.

14. L. Snyder. Formal models of capability-based protection systems. *IEEE Transactions on Computers*, 30:172, 1981.
15. M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems*. J. Wiley & Sons, Chichester, 2004.
16. Tuva Hassel Stang, Fahimeh Pourbayat, Mark Burgess, Geoffrey Canright, Kenth Engø, and Åsmund Weltzien. Archipelago: A network security analysis tool. In *Proceedings of The 17th Annual Large Installation Systems Administration Conference (LISA 2003)*, San Diego, California, USA, October 2003.
17. G. Canright and K. Engø-Monsen. A natural definition of clusters and roles in undirected graphs. *Science of Computer Programming*, 53:195, 2004.
18. M. Burgess, G. Canright, and K. Engø. A graph theoretical model of computer security: from file access to social engineering. *International Journal of Information Security*, 3:70–85, 2004.
19. R. Ortalo. A flexible method for information system security policy specifications. *Lecture Notes on Computer Science*, 1485:67–85, 1998.
20. J. Glasgow, G. MacEwan, and P. Panagaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10:226–264, 1992.
21. E. Lupu and M. Sloman. Conflict analysis for management policies. In *Proceedings of the Vth International Symposium on Integrated Network Management IM'97*, pages 1–14. Chapman & Hall, May 1997.
22. B.F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.
23. H. Prakken and M. Sergot. Dyadic deontic logic and contrary-to-duty obligations. In *Defeasible Deontic logic: Essays in Nonmonotonic Normative Reasoning*, volume 263 of *Synthese library*. Kluwer Academic Publisher, 1997.
24. S.A. Kripke. Semantical considerations in modal logic. *Acta Philosophica Fenica*, 16:83–94, 1963.
25. S. Fagernes and M. Burgess. The effects of 'tit for tat' policy for rejecting 'spam' or denial of service floods. In *Proceedings of the 4th System Administration and Network Engineering Conference (SANE 2004)*, 2004.
26. M. Burgess and S. Fagernes. Pervasive computing management ii: Voluntary cooperation. *IEEE eTransactions on Network and Service Management*, page (submitted).
27. A.K. Bandara, E.C. Lupu, J. Moffett, and A. Russo. A goal-based approach to policy refinement. In *Proceedings of the 5th IEEE Workshop on Policies for Distributed Systems and Networks*, 2004.
28. A.K. Bandara, E.C. Lupu, J. Moffett, and A. Russo. Using event calculus to formalise policy specification and analysis. In *Proceedings of the 4th IEEE Workshop on Policies for Distributed Systems and Networks*, 2003.
29. A.L. Lafuente and U. Montanari. Quantitative mu-calculus and ctl defined over constraint semirings. *Electronic Notes on Theoretical Computing Systems QAPL*, pages 1–30, 2005.

Towards Automated Deployment of Built-to-Order Systems

Akhil Sahai[1], Calton Pu[2], Gueyoung Jung[2],
Qinyi Wu[2], Wenchang Yan[2], and Galen S. Swint[2]

[1] HP Laboratories, Palo-Alto, CA
akhil.sahai@hp.com
[2] Center for Experimental Research in Computer Systems,
College of Computing, Georgia Institute of Technology,
801, Atlantic Drive, Atlanta, GA 30332
{calton, helcyon1, qxw, wyan, galen.swint}@cc.gatech.edu

Abstract. End-to-end automated application design and deployment poses a significant technical challenge. With increasing scale and complexity of IT systems and the manual handling of existing scripts and configuration files for application deployment that makes them increasingly error-prone and brittle, this problem has become more acute. Even though design tools have been used to automate system design, it is usually difficult to translate these designs to deployed systems in an automated manner due to both syntactic obstacles and the synchronization of multiple activities involved in such a deployment. We describe a generic process of automated deployment from design documents and evaluate this process for 1, 2, and 3-tier distributed applications.

1 Introduction

New paradigms, such as autonomic computing, grid computing and adaptive enterprises, reflect recent developments in industry [1, 2, 3] and research [4]. Our goal is to create "Built-to-Order" systems that operate in these new computing environments. This requires easy and automated application design, deployment, and management tools to address their inherent complexity. We must support creating detailed designs from which we can deploy systems. These designs, in turn, are necessarily based on user requirements that take into account both operator and technical capability constraints. Creating design in an automated manner is a hard problem in itself. Quartermaster Cauldron [5], addresses the challenge by modeling system components with an object-oriented class hierarchy, the CIM (Common Information Model) metamodel, and embedding constraints on composition within the models as policies. Then, Cauldron uses a constraint satisfaction approach to create system designs and deployment workflows. However, these workflows and designs are expressed in system-neutral Managed Object Format (MOF).

MOF workflows typically involve multiple systems and formats that have to be dealt with in order to deploy a complex system. For example, deploying a three-tier e-commerce solution in a virtualized environment may involve interactions with blade

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 109 – 120, 2005.

servers, VMWare/Virtual Servers, multiple operating systems, service containers for web servers, application servers, databases, before, finally, executing clients scripts. This problem of translating generic design in a system independent format (*e.g.*, MOF) to the multiple languages/interfaces demanded by the system environment is thus nontrivial.

The main contribution of the paper is a generic mechanism for translating design specifications written in a system independent format into multiple and varied deployment environments. To achieve this generic translation, we use an XML based intermediate representation and a flexible code generation method [6, 7] to build an extensible translator, the Automated Composable Code Translator (ACCT). Translation between the two models is non-trivial and significant result for two reasons. First, the models are quite dissimilar in some aspects; the translation is not a straightforward one-to-one mapping. For example, we describe the translation between significantly different workflow models in Section 0. Second, the ACCT design is deliberately generic to accommodate the multiple input and output formats encountered in multiple design and d environments. ACCT accepts MOF-based design specifications of CIM instance models and converts them into input specifications for SmartFrog, a high-level deployment tool [8]. SmartFrog uses on a high-level specification language and Java code to install, execute, monitor, and terminate applications. The generic architecture supporting multiple input/output formats is described elsewhere [6, 7].

2 Automated Design and Automated Deployment

2.1 Automated Design Environment

At the highest level of abstraction, automated design tools offer streamlined and verified application creation. Quartermaster is an integrated tool suite built around MOF to support automated design of distributed applications at this high level of abstraction [9, 10]. Cauldron, one of its key components, supports applying policies and rules at design-time to govern composition of resources. Cauldron's constraint satisfaction engine can generate system descriptions that satisfy these administrative and technical constraints. In this paper, we concentrate on deployment constraints for distributed applications. Since each component of an application often depends on prior deployment of other components or completion of other components' work, deployment is non-trivial.

To model deployment, we use a MOF Activity comprised of a number of subactivities. Each of these activities has a set of constraints to meet before execution and also parameters that must receive values. At design time, Cauldron generates configuration templates and also pairwise deployment dependencies between deployment activities. Between any pair of activities, there are four possible synchronization dependencies.

SS (Start-Start) – activities must start together; a symmetric, transitive dependency.

FF (Finish-Finish) –activities must finish together (synchronized); also a symmetric, transitive dependency.

FS (Finish-Start) – predecessor activity must complete before the successor activity is started, *i.e.*, sequential execution. This dependency implies a strict ordering, and the MOF must assign either the antecedent or the dependant role to each activity component.

SF (Start-Finish) – predecessor activity is started before the successor activity is finished. Similar observations on its properties follow as from FS. (As an SF example, consider producer-consumer relationships in which the producer must create a communication endpoint before the consumer attempts attachment.)

Cauldron, however, is solely a design tool and provides no deployment tools, which require software that initiate, monitor, and kill components in a distributed environment.

2.2 Automated Deployment Environment

Automated deployment tools serve to ameliorate the laborious process of preparing, starting, monitoring, stopping, and even post-execution clean-up of distributed, complex applications. SmartFrog is an open-source, LGPL framework that supports such service deployment and lifecycle management for distributed Java applications [11, 12]; it has been used on the Utility Computing model for deploying rendering code on demand and has been ported to PlanetLab [13]. Expertise gained applying SmartFrog to grid deployment [14] is being used in the CDDLM standardization effort currently underway.

Conceptually, SmartFrog comprises 1) a component model supporting application-lifecycle operations and workflow facilities, 2) a specification language and validator for these specifications, and 3) tools for distribution, lifecycle monitoring, and control. The main functionalities of SmartFrog are as follows:

Lifecycle operations – SmartFrog wraps deployable components and transitions them through their life phases: *initiate*, *deploy*, *start*, *terminate,* and *fail.*

Workflow facilities – Allows flexible control over configuration dependencies between components to create workflows. Examples: *Parallel*, *Sequence*, and *Repeat.*

SmartFrog runtime – Instantiates and monitors components; provides security. The runtime manages interactions between daemons running on remote hosts. It provides an event framework to send and receive events without disclosing component locations.

SmartFrog's specification language features data encapsulation, inheritance, and composition which allows system configurations to be incrementally declared and customized. In practice, SmartFrog needs three types of files to deploy an application:

1. Java `interface` definitions for components. These serve analogously to the interface exposure role of the C++ header file and `class` construct.
2. Java source files that implement components as objects. These files correspond one-to-one with the above SmartFrog component descriptions.
3. A single instantiation and deployment file, in a SmartFrog specific language, defining the parameters and proper global deployment order for components.

2.3 Translating Between Design Specifications and Deployment Specifications

This section describes ACCT, our extensible, XML-based tool that translates generic design specifications into fully parameterized, executable deployment specifications. First, we describe ACCT's design and the implementation and then the mapping approach needed to resolve mismatches between the design tool output (MOF) and deployment tool input (SmartFrog).

There are several obstacles to translating Cauldron to SmartFrog. First, there is the syntax problem; Cauldron generates MOF, but SmartFrog requires a document in its own language syntax as well as two more types supporting of Java source code. Obviously, this single MOF specification must be mapped to three kinds of output files, but neither SmartFrog nor Quartermaster supports deriving Java source from the design documents. Finally, Cauldron only produces *pairwise* dependencies between deployment activities; SmartFrog, on the other hand, needs dependencies over the entire set of deployment activities to generate a deployment workflow for the system.

In ACCT, code generation is built around an XML document which is compiled from a high-level human-friendly specification language (MOF) and then transformed using XSLT. So far, this approach has been applied to a code generation system for information flow architectures and has shown several benefits including support for rapid development, extensibility to both new input and output languages, and support for advanced features such as source-level aspect weaving. These advantages mesh well with ACCT's goals of multiple input languages and multiple output languages, and SmartFrog deployments, in fact, require ACCT to generate two different output formats.

The code translation process consists of three phases which are illustrated in Fig. 1. In the first phase, MOF-to-XML, ACCT reads MOF files and compiles them into a single XML specification, XMOF, using our modification of the publicly available WBEM Services' CIM-to-XML converter [15].

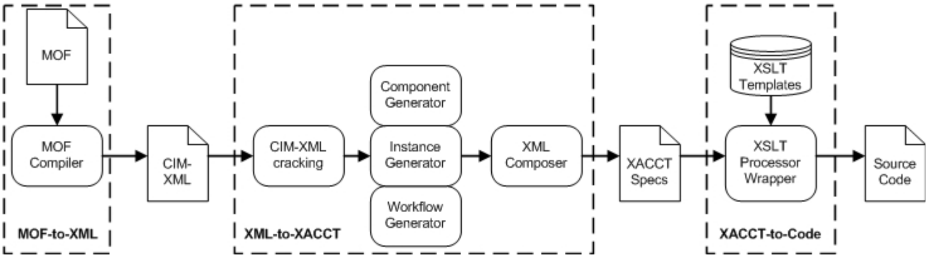


Fig. 1. The ACCT code generator

In phase two, XML-to-XACCT, XMOF is translated into a set of XACCT documents, the intermediate XML format of the ACCT tool. During this transformation, ACCT processes XMOF in-memory as a DOM tree and extracts three types of Cauldron-embedded information: components, instances, and deployment workflow. Each data sets is processed by a dedicated code generator written in Java. The component generator creates an XML component description, the instance generator produces a set of attributes and values, as XML, for deployed components, and the workflow generator computes a complete, globally-ordered workflow expressed in XML. (We will describe the workflow construction in more detail later.) These generated structures are passed to an XML composer which performs rudimentary type checking (to ensure instances are only present if there is also a class), and re-aggregates the XML fragments back into a whole XML documents. This may result in multiple XACCT component description documents, but there is only one instantiation+workflow document which contains the needed data for a single deployment.

Finally, in the third phase ACCT forwards each XACCT component description, instantiation, and workflow document to the XSLT processor. The XSLT templates detect the XACCT document type and generate the appropriate files (SmartFrog or Java) which are written to disk.

XACCT allows components, configurations, constraints, and workflows from input languages of any resource management tool to be described in an intermediate representation. Once an input language is mapped to XACCT, the user merely creates an XSLT template for the XML-to-XACCT phase to perform the final mapping of the XACCT to a specific target language. Conversely, one need only add a new template to support new target languages from an existing XACCT document.

Purely syntactic differences between MOF and SmartFrog's language can be resolved using solely XSLT, and the first version of ACCT was developed on XSLT alone. However, because the XSLT specification version used for ACCT had certain limitations, we incorporated Java pre- and post- processing stages. This allowed us to compute the necessary global SmartFrog workflows from the MOF partial workflows and to create multiple output files from a single ACCT execution.

Overall system ordering derives from the Cauldron computed partial synchronizations encoded in the input MOF. As mentioned in Section 0, MOF defines four types of partial synchronization dependencies: SS, FF, SF, and FS. To describe the sequential and parallel ordering of components which SmartFrog requires, these partial dependencies are mapped via an event queue model with an algorithm that synchronizes activities correctly. It is helpful to consider the process as that of building a graph in which each component is a node and each dependency is an edge. Each activity component has one associated EventQueue containing list of actions:

Execute – execute a specific sub-component.

EventSend – send a specific event to other components. This may accept a list of destination components.

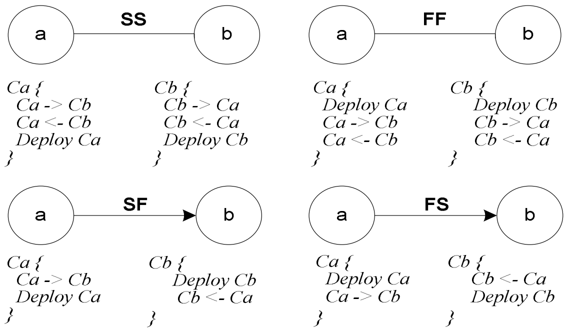
OnEvent – the action to wait for an incoming event. This may wait on events from multiple source components. It is the dual of EventSend.

Terminate – the action to remove the EventQueue.

Table 1. Possible event dependencies between components

C	A component, C.
$C_a \rightarrow C_b$	Component C_a sends event to C_b.
$C_a \leftarrow C_b$	Component C_a waits for event from C_b.
$C_a — C_b$	Components *must* perform action together

Given this model, any two components may have one of three synchronization relations, as shown in Table 1. Fig. 2 applies these synchronization semantics to the pairwise MOF relationships. In SS, two activity components are blocked until each event receives a “start” event from the other. In ACCT, this translates to entries in the EventQueues to send and then wait for an event from the peer component. The FF scenario is handled similarly. In SF, since C_b’s activity must be finished after C_a starts to deploy, C_b is blocked in its EventQueue until C_a’s “start” is received at C_b. In FS, since C_b may deploy only after C_a completes its task, C_b blocks until a “finished” event from C_a is received at C_b. (For now, we assume the network delay between two components is negligible.)



```
Ca {            Cb {            Ca {            Cb {
 Ca -> Cb        Cb -> Ca        Deploy Ca       Deploy Cb
 Ca <- Cb        Cb <- Ca        Ca -> Cb        Cb -> Ca
 Deploy Ca       Deploy Cb       Ca <- Cb        Cb <- Ca
}               }               }               }

Ca {            Cb {            Ca {            Cb {
 Ca -> Cb        Deploy Cb       Deploy Ca       Cb <- Ca
 Deploy Ca       Cb <- Ca        Ca -> Cb        Deploy Cb
}               }               }               }
```

Fig. 2. Diagrams and dependency formulations of SS, FF, SF, and FS

Fig. 3 illustrates the XACCT for the FS dependency. The SS and FF operations are parallel deployment activities while SF and FS represent sequential deployment activities.

The exact content of each EventQueue depends on its dependencies to all other activity components. Since each activity component frequently has multiple dependencies, we devised an algorithm to calculate EventQueue contents.

```
<Instance Name="Ca" Class="Activity">
  <Workflow>
    <Work Name="--" Type="Execute">
    </Work>
    <Work Name="--" Type="EventSend">
      <To>Cb</To>
    </Work>
    <Work Name="--" Type="Terminator">
     Ca</Work>
  </Workflow>
</Instance>
<Instance Name="Cb" Class="Activity">
  <Workflow>
    <Work Type="OnEvent">
      <From>Ca</From>
    </Work>
    <Work Name="--" Type="Execute">
    </Work>
    <Work Name="--" Type="Terminator">
     Cb</Work>
  </Workflow>
</Instance>
```

Fig. 3. XACCT snippet for the FS dependency

The algorithm visits each activity component, C_i, in the XMOF to build a global action list. If a dependency of the component is a parallel dependency (SS or FF), then the algorithm transitively checks for dependencies of the same type on related activity components until it finds no more parallel. For example, if there is a dependency in which C_i is SS with C_j, and C_j is also SS with C_k but FF (a different parallel dependency) with C_m, it records only “C_j and C_k” as SS on its action list before proceeding to check component C_{i+1}. If it is a sequential dependency (FS or SF), the algorithm adds the dependency to the global action list and proceeds to component i+1. That is, if C_i has FS with C_j, and C_j has FS with C_k, only the pairwise relation “C_i and C_j with FS” is entered into the global action list before proceeding to C_{i+1}.

Then the algorithm implements deadlock avoidance by enforcing a static order of actions for each activity component C_i based on the component’s role, antecedent or dependant, in each relationship. The algorithm checks the six possible combinations of roles and dependencies as follows:

1. If C_i participates as a dependant in any FS relationship, then it adds one OnEvent action to the EventQueue per FS-Dependancy.
2. If C_i has any SS dependencies, then it adds all needed EventSend and OnEvent actions to the EventQueue.
3. If C_i functions as antecedent in SF dependencies, then per dependency it adds an EventSend action to the EventQueue followed by a single Execute action.
4. If C_i participates as a dependant in an SF dependency, then one OnEvent action per dependency is added to the EventQueue.
5. If C_i has any FF dependencies, it adds all EventSend and OnEvent actions to the EventQueue.
6. Finally, if C_i serves as an antecedent roles with FS, then it adds one EventSend action per FS occurrence.

Finally, the workflow algorithm appends the "Terminate" action to each C_i's EventQueue.

XACCT captures the final workflow in a set of per-component EventQueues, which ACCT then translates to the input format of the deployment system (*i.e.*, SmartFrog). The Java source code generated by ACCT is automatically compiled, packaged into a `jar` file, and integrated into SmartFrog using its class loader. An HTTP server functions as a repository to store scripts and application source files. Once a SmartFrog description is fed to the SmartFrog daemon, it spawns one thread for each activity in the workflow. Subsequent synchronization among activities is controlled by EventQueues.

3 Demo Application and Evaluation

We present in this section how Cauldron-ACCT-SmartFrog toolkit operates from generating the system configurations and workflow, translating both into the input of SmartFrog, and then automatically deploying distributed applications of varying complexity. In the subsection 0, we describe 1-, 2-, and 3-tier example applications and system setup employed for our experiments. Following that, we evaluate our toolkit by comparing the deployment execution time of SmartFrog and automatically generated deployment code to manually written deployment scripts.

3.1 Experiment Scenario and Setup

We evaluated our translator by employing it on 1-, 2-, and 3-tier applications. The simple 1- and 2-tier applications provide baselines for comparing a generated SmartFrog description to hand-crafted scripts. The 3-tier testbed comprises web, application, and database servers; it is a small enough size to be easily testable, but also has enough components to illustrate the power of the toolkit for managing complexity. Table 2, below, lists each scenario's components.

Table 2. Components of 1-, 2-, and 3-tier applications

Scenario	Application	Components
1-tier	Static web page	Web Server : Apache 2.0.49
2-tier	Web Page Hit Counter	Web Server : Apache 2.0.49 App. Server : Tomcat 5.0.19 Build System: Apache Ant 1.6.1
3-tier	iBATIS JPetStore 4.0.0	Web Server : Apache 2.0.49 App. Server : Tomcat 5.0.19 DB Server : MySQL 4.0.18 DB Driver : MySQL Connector to Java 3.0.11 Build System : Apache Ant 1.6.1 Others : DAO, SQLMap, Struts

We installed SmartFrog 3.04.008_beta on four 800 MHz dual-processor Dell Pentium III machines running RedHat 9.0; one SmartFrog daemon runs on each host.

In the 1-tier application, we deploy Apache as a standalone web server, and confirmed successful deployment by visiting a static web page. The evaluation used two machines: the first for the web server and a second to execute the generated SmartFrog workflow.

In the 2-tier Hit Counter application, we used Apache and the Tomcat application server with Ant. Each tier specified for deployment to a separate host. To verify the 2-tier deployment, we visited the web page multiple times to ensure it recorded page hits. The application simply consists of a class and a `jsp` page. The 2-tier evaluation required three machines. As in the 1-tier test, we used one machine to run the deployment script; then, we dedicated one machine to each deployed tier (Apache; Ant and Tomcat).

Finally, the 3-tier application was the iBATIS JPetStore, a ubiquitous introduction to 3-tier programming. In the 3-tier application evaluation, we used four machines. Again, we dedicated one machine for each tier (Apache; Tomcat, JPetStore, Ant, MySQL Driver, Struts; MySQL DB) and used a fourth machine to run the SmartFrog workflow.

Fig. 4 illustrates the dependencies of components in each testbed. We consider three types of dependencies in the experiment; installation dependency, configuration dependency, and activation dependency. The total number of dependencies in each testbed is used as the level of the complexity. In Figure 6, 1-, 2-, and 3-tier testbeds are considered as simple, medium, and complex cases respectively. Intuitively, the installation, configuration, and activation dependencies of each component in each testbed must be sequenced. For instance, the Apache configuration must start after Apache installation completes, and Apache activation must start after Apache configuration completes. (For space, we have omitted these dependencies from the figure.)

We modeled 1-, 2-, and 3-tier applications in Quartermaster with and Cauldron module created the configurations and deployment workflows. The resultant MOF

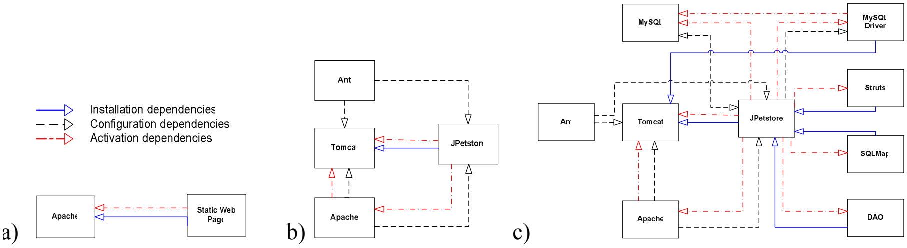


Fig. 4. Dependency diagrams of (a) 1-tier, (b) 2-tier, and (c) 3-tier application

files were fed into ACCT and yielded a set of Java class files, SmartFrog component descriptions, and a SmartFrog instances+workflow specification for each application tested. Fig. 6. on the following page illustrates the transformation process as ACCT translates the MOF file of the 3-tier application to intermediate XACCT and then finally to a SmartFrog description. For demonstration, we highlight the FS dependency between the Tomcat installation and MySQLDriver installation and how this information is carried through the transformation process.

3.2 Experimental Result

The metric we choose for evaluating the 1-, 2-, and 3-tier testbeds is deployment execution time as compared to manually written scripts. We executed SmartFrog and scripts 30 times each for each tier application and report the average. Fig. 5 shows that for simple cases (1- and 2-tier) SmartFrog took marginally longer when compared to the scripts based approach because SmartFrog daemons running in the Java VM impose extra costs when loading Java classes or engaging in RMI communication. The trend favors SmartFrog as the time penalty of the medium case becomes less (in absolute and relative terms) and for the complex case, SmartFrog took less time than the scripts based approach.

In the complex case, SmartFrog was able to exploit concurrency between application components since it had a computed workflow. The simple and medium cases contain fewer concurrent dependencies than the 3-tier case. Nevertheless, in all cases our toolkit retains the important advantage of an automatically generated workflow, while in scripts based approach, system administrators must manually control the order of installing, configuration, and deployment.

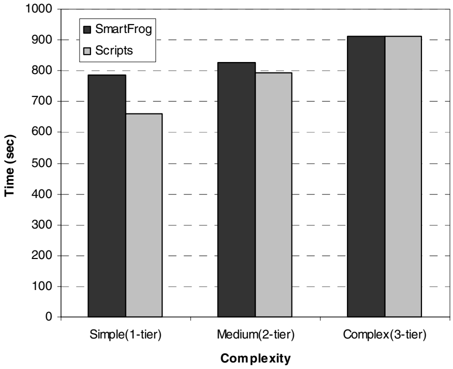
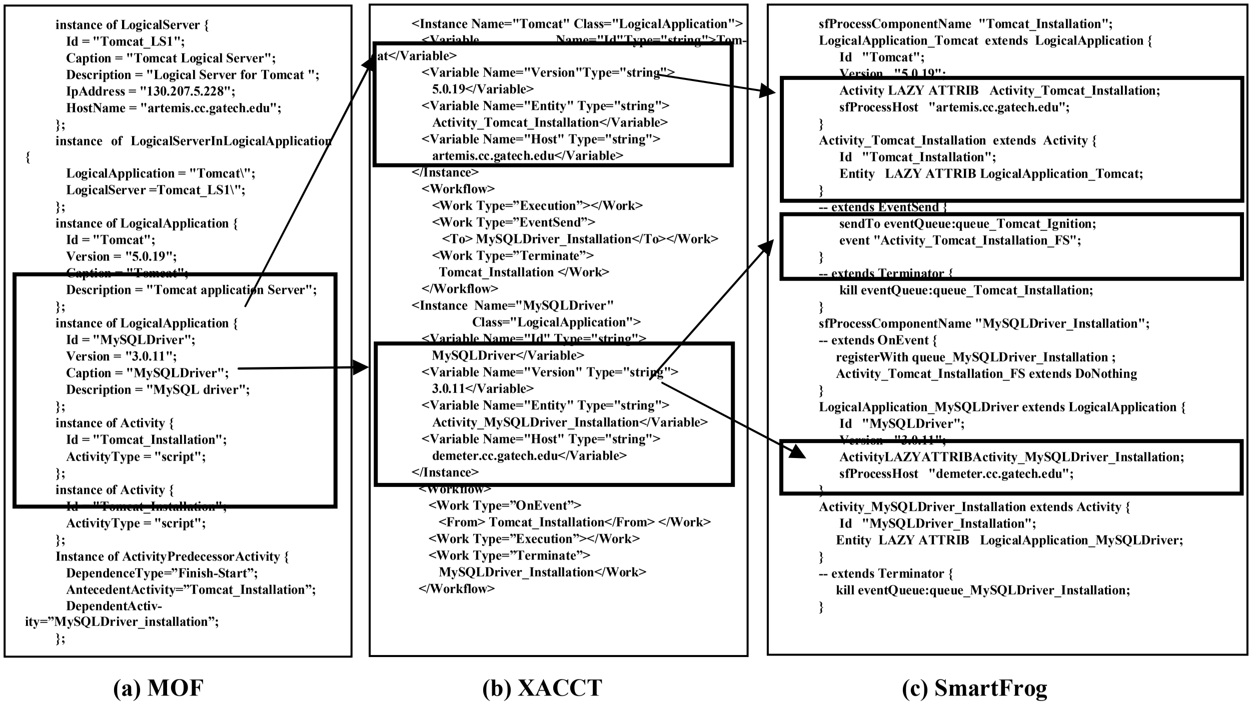


Fig. 5. Deployment Time using SmartFrog and scripts as a function of the complexity



```
        instance of LogicalServer {
          Id = "Tomcat_LS1";
          Caption = "Tomcat Logical Server";
          Description = "Logical Server for Tomcat ";
          IpAddress = "130.207.5.228";
          HostName = "artemis.cc.gatech.edu";
        };
        instance  of  LogicalServerInLogicalApplication
{
          LogicalApplication = "Tomcat\";
          LogicalServer =Tomcat_LS1\";
        };
        instance of LogicalApplication {
          Id = "Tomcat";
          Version = "5.0.19";
          Caption = "Tomcat";
          Description = "Tomcat application Server";
        };
        instance of LogicalApplication {
          Id = "MySQLDriver";
          Version = "3.0.11";
          Caption = "MySQLDriver";
          Description = "MySQL driver";
        };
        instance of Activity {
          Id = "Tomcat_Installation";
          ActivityType = "script";
        };
        instance of Activity {
          Id = "Tomcat_Installation";
          ActivityType = "script";
        };
        Instance of ActivityPredecessorActivity {
          DependenceType="Finish-Start";
          AntecedentActivity="Tomcat_Installation";
          DependentActiv-
ity="MySQLDriver_installation";
        };
```

(a) MOF

```
   <Instance Name="Tomcat" Class="LogicalApplication">
     <Variable            Name="Id"Type="string">Tom-
at</Variable>
     <Variable Name="Version"Type="string">
       5.0.19</Variable>
     <Variable Name="Entity" Type="string">
       Activity_Tomcat_Installation</Variable>
     <Variable Name="Host" Type="string">
       artemis.cc.gatech.edu</Variable>
   </Instance>
     <Workflow>
       <Work Type="Execution"></Work>
       <Work Type="EventSend">
         <To> MySQLDriver_Installation</To></Work>
       <Work Type="Terminate">
        Tomcat_Installation </Work>
     </Workflow>
   <Instance  Name="MySQLDriver"
              Class="LogicalApplication">
     <Variable Name="Id" Type="string">
       MySQLDriver</Variable>
     <Variable Name="Version" Type="string">
       3.0.11</Variable>
     <Variable Name="Entity" Type="string">
       Activity_MySQLDriver_Installation</Variable>
     <Variable Name="Host" Type="string">
       demeter.cc.gatech.edu</Variable>
   </Instance>
     <Workflow>
       <Work Type="OnEvent">
         <From> Tomcat_Installation</From> </Work>
       <Work Type="Execution"></Work>
       <Work Type="Terminate">
         MySQLDriver_Installation</Work>
     </Workflow>
```

(b) XACCT

```
sfProcessComponentName  "Tomcat_Installation";
LogicalApplication_Tomcat  extends  LogicalApplication {
    Id   "Tomcat";
    Version   "5.0.19";
    Activity LAZY ATTRIB   Activity_Tomcat_Installation;
    sfProcessHost   "artemis.cc.gatech.edu";
}
Activity_Tomcat_Installation  extends  Activity {
    Id   "Tomcat_Installation";
    Entity   LAZY ATTRIB LogicalApplication_Tomcat;
}
-- extends EventSend {
    sendTo eventQueue:queue_Tomcat_Ignition;
    event "Activity_Tomcat_Installation_FS";
}
-- extends Terminator {
    kill eventQueue:queue_Tomcat_Installation;
}
sfProcessComponentName "MySQLDriver_Installation";
-- extends OnEvent {
   registerWith queue_MySQLDriver_Installation ;
   Activity_Tomcat_Installation_FS extends DoNothing
}
LogicalApplication_MySQLDriver extends LogicalApplication {
    Id   "MySQLDriver";
    Version   "3.0.11";
    ActivityLAZYATTRIBActivity_MySQLDriver_Installation;
    sfProcessHost   "demeter.cc.gatech.edu";
}
Activity_MySQLDriver_Installation extends Activity {
    Id   "MySQLDriver_Installation";
   Entity  LAZY ATTRIB   LogicalApplication_MySQLDriver;
}
-- extends Terminator {
   kill eventQueue:queue_MySQLDriver_Installation;
}
```

(c) SmartFrog

Fig. 6. (a) MOF, (b) Intermediate XML, and (c) SmartFrog code snippets. The solid line box indicates the FS workflow between Tomcat and MySQLDriver applications. Others indicate configurations. Clearly, MOF offers superior understandability for a deployment scenario as compared to the SmartFrog specification. As Vanish et al showed in [16], automating deployment via SmartFrog, for which we generate code, is generally superior in performance and more maintainable when compared to manual or ad hoc scripted solutions.

4 Related Work

Recent years have seen the advent of wide-ranging resource management systems. For e-business, OGSA Grid Computing [17] aims to provide services within an on-demand data center infrastructure. IBM's Autonomic Computing Toolkit [1], the HP Utility Data Center [18] and Microsoft's DSI initiative [3] are examples of this. The distinction of our toolkit, however, is that Cauldron logic and a theorem prover to meet resource allocation constraints. There are several efforts related to specifying conditions and actions for policies, *e.g.*, CIM [19] and PARLAY [20]. However, to the best of our knowledge, none of them have used a constraint satisfaction approach for automatic resource construction.

Another trend is deployment automation tools. CFengine [22] provides rich facilities for system administration and is specifically designed for testing and configuring software. It defines a declarative language so that the transparency of a configuration program is optimal and management is separate from implementation. Nix [21] is another popular tool used to install, maintain, control, and monitor applications. It is capable of enforcing reliable specification of component and support for multiple version of a component. However, since Nixes does not provide automated workflow mechanism, users manually configure the order of the deployments. For deployment of a large and complicated application, it becomes hard to use Nixes. By comparison, SmartFrog provides control flow structure and event mechanism to support flexible construction of workflow.

The ACCT translator adopts the Clearwater architecture developed for the Infopipe Stub Generator + AXpect Weaver (ISG) [6, 7]. Both ACCT and ISG utilize an XML intermediate format that is translated by XSLT to target source code. Unlike ACCT, however, ISG is designed for creating information flow system code. There are other commercial and academic translation tools, like MapForce [23] and CodeSmith [24]. Similar to ISG, they target general code generation and do not support deployment workflows.

5 Conclusion

We outlined an approach for Automated Deployment of complex distributed applications. Concretely, we described in detail the ACCT component (Automated Composable Code Translator) that translates Cauldron output (in XMOF format) into a SmartFrog specification that can be compiled into Java executables for automated deployment. ACCT performs a non-trivial translation, given the differences between the XMOF and SmartFrog models such as workflow dependencies. A demonstration application (JPetStore) illustrates the automated design and implementation process and translation steps, showing the increasing advantages of such automation as the complexity of the application grows.

Acknowledgement. This work was partially supported by NSF/CISE IIS and CNS divisions through grants IDM-0242397 and ITR-0219902, DARPA ITO and IXO through contract F33615-00-C-3049 and N66001-00-2-8901, and Hewlett-Packard. We thank our anonymous reviewers and M. Feridun for their very helpful comments.

References

1. IBM Autonomic Computing. http://www.ibm.com/autonomic.
2. SUN N1. http://www.sun.com/software/n1gridsystem/.
3. Microsoft DSI. http://www.microsoft.com/windowsserversystem/dsi/.
4. Global Grid Forum. http://www.ggf.org.
5. Sahai, A., Singhal, S., Joshi, R., Machiraju, V.: Automated Policy-Based Resource Construction in Utility Computing Environments. NOMS, 2004.
6. Swint, G. and Pu, C.: Code Generation for WSLAs using AXpect. 2004 IEEE International Conference on Web Services. San Diego, 2004.
7. Swint, G., Pu, C., Consel, C., Jung, G., Sahai, A., Yan, W., Koh, Y., Wu, Q.: Clearwater - Extensible, Flexible, Modular Code Generation. 20th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2005.
8. SmartFrog. http://www.hpl.hp.com/research/smartfrog/.
9. Salle, M., Sahai, A., Bartolini, C., Singhal, S.: A Business-Driven Approach to Closed-Loop Management. HP Labs Technical Report HPL-2004-205, November 2004.
10. Sahai, Akhil, Sharad Singhal, Rajeev Joshi, Vijay Machiraju: Automated Generation of Resource Configurations through Policies. IEEE Policy, 2004.
11. Goldsack, Patrick, Julio Guijarro, Antonio Lain, Guillaume Mecheneau, Paul Murray, Peter Toft.: SmartFrog: Configuration and Automatic Ignition of Distributed Applications. HP Openview University Association conference, 2003.
12. Smartfrog open source directory. http://sourceforge.net/projects/smartfrog.
13. Peterson, Larry, Tom Anderson, David Culler, and Timothy Roscoe: A Blueprint for Introducing Disruptive Technology. PlanetLab Tech Note, PDN-02-001, July 2002.
14. CDDLM Foundation Document. http://www.ggf.org/Meetings/ggf10/GGF10%20Documents/CDDLM_Foundation_Document_v12.pdf.
15. WBEM project. http://wbemservices.sourceforge.net.
16. Talwar, Vanish, Dejan Milojicic, Qinyi Wu, Calton Pu, Wenchang Yan, and Gueyoung Jung: Comparison of Approaches to Service Deployment. ICDCS 2005.
17. Foster, Ian, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Globus Project, 2002.
18. HP Utility Data Center. http://www.hp.com/products1/promos/adaptive_enterprise/us/utility.html.
19. DMTF-CIM Policy. http://www.dmtf.org/standards/cim/cim_schema_v29.
20. PARLAY Policy Management. http://www.parlay.org/specs/.
21. Dolstra, Eelco, Merijn de Jonge, and Eelco Visser: Nix: A Safe and Policy-free System for Software Deployment. 18th Large Installation System Administration Conference, 2004.
22. Cfengine. http://www.cfengine.org/.
23. Altova Mapforce. http://www.altova.com/products_mapforce.html.
24. Codesmith. http://www.ericjsmith.net/codesmith.

A Generic Model and Architecture for Automated Auditing

Hasan[1] and Burkhard Stiller[1,2]

[1] Computer Engineering and Networks Laboratory TIK, ETH Zürich, Switzerland
{hasan, stiller}@tik.ee.ethz.ch
[2] Computer Science Department IFI, University of Zürich, Switzerland
stiller@ifi.unizh.ch

Abstract. Research has been performed in areas of auditing, a.o. security auditing, compliance auditing, financial auditing. In order to increase the efficiency of and to allow for continuous auditing, auditing tasks must be automated, which is only possible if audit data are available digitally and suitable algorithms exist. Different areas of auditing follow different objectives, thus require different detailed tasks to be performed, yet they share a common auditing model. This is based on the consideration that in general auditing deals with the evaluation or examination of facts against a set of compliance specifications. The objective of this paper is to develop a generic model and architecture for automated auditing, thus providing the basis for the development of auditing work for specific applications. To show its general applicability, the proposed model is applied to different areas including Service Level Agreement (SLA) compliance verification and Intrusion Detection Systems. A full-fledged example is discussed showing in detail how the generic architecture is applied to the SLA compliance verification.

1 Introduction

Auditing is a widely applied concept for investigating the adequacy of a system against a set of requirements. Traditional areas of auditing comprise amongst others financial audits, compliance audits with respect to laws and regulations, performance audits, and quality audits. The wide use of the Internet by research and government institutions, companies, and individuals, as well as the commercialization of Internet services have opened up a new and important area of auditing including information system security audits and Service Level Agreements (SLA) compliance audits.

The development and deployment of the Internet has bridged the path to the information era, where information becomes a vital resource. Usually information, in form of digital objects, is stored in a computer system, which is connected to the Internet. Since access to the Internet can be gained by anyone, the Internet is subject to attacks. To cope with attacks, various Intrusion Detection Systems (IDS) have been developed. An IDS is a security audit tool to reveal unauthorized access attempts.

Today, many business relationships between a service provider and a customer are formally defined in terms of SLAs, which specify contractual commitments of a provider on which services and with which measurable quality level the provider will furnish [4]. The committed quality level of a service is specified in a set of Service Level

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 121 – 132, 2005.

Objectives (SLOs) in form of service metrics, threshold values, and tolerances [16]. Failure to perform contractual obligation is a contract violation. In order to detect this violation all related business transactions have to be accounted for and audited. Furthermore, for Internet services many communication protocols have been standardized and policy-based network management systems have been developed, where auditing can be useful to find non-compliances in protocol implementations and policy decisions.

Traditionally, auditing is accomplished by human auditors through a manual audit of paper documentation, which is very time-consuming. This leads to the question whether, at least, parts of the auditing process can be automated. Automation is easier to achieve in those areas of auditing, where materials are available in a structured digital format. Auditing deals with varieties of information, which often have a complex interrelation. In many cases, auditing requires a strong analytical skill of the auditor, and the transfer of this skill to an automated auditor poses a challenge. The two main reasons for automated auditing are a high degree of efficiency in processing a large number of audit data and earlier detection of non-compliances through continuous auditing.

Thus, the key goal of this paper is to develop the common denominator of auditing tasks in different areas and to design a generic model and architecture for automated auditing. Based on this model and architecture a framework for various auditing purposes is derived to minimize further efforts in implementing specific auditing applications.

The remainder of the paper is organized as follows. Section 2 discusses related work from different auditing areas. While Section 3 presents a generic auditing model, Section 4 develops the generic architecture. The application of the model and architecture is presented in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

Most of existing approaches in auditing are dedicated to specific objectives. Although some proposals show to a certain extent a general approach, a generic auditing model and architecture—to the best knowledge of the authors at the moment of writing—is not yet available. Hence, this section describes major related work in specific areas of auditing, including security auditing and SLA compliance verification.

IDSs have been a research area of security auditing since the beginning of the 1980s. In 1987 [5] presented a model for a real-time intrusion detection expert system, which provides the basis for many IDSs. The model can be regarded as a rule-based pattern matching system and contains the following six main components: subjects, objects, audit records, profiles, anomaly records, and activity rules. In this model actions performed by subjects on objects are essential, which is valid for an IDS, but not in all areas of auditing. For example, in the case of SLA compliance verifications services delivered by objects to subjects are relevant. Therefore, the relation between subjects and objects is not described in a generic auditing model. Furthermore, profiles in this model, which describe a normal behavior of subjects on objects, are updated based on audit records. Here, profiles define the specifications to be met. However, general auditing does not modify compliance specifications so that they are met by the normal behavior.

The architecture of a general IDS comprises of event generators within a target system, analysis engines, and a response unit [11]. Components can be distributed; analysis engines can form a hierarchy. Recent architectures use autonomous and mobile agents to perform the task of event generators and analysis engines. Distribution of IDS components happens within a single administrative domain, whereas general auditing can stretch across multiple ones.

Academia and industries have performed research and developed prototypes or even products for SLA management and monitoring of SLA compliances. However, most of these approaches are dedicated to specific services, e.g., web services, or a certain set of SLA parameters, e.g., availability, round-trip time, and response time [3], [6], [7], [9], [10], [15]. The IBM's Web Service Level Agreement (WSLA) Framework shows a general concept for SLA management and defines a language to specify SLAs, focusing on web services [10]. It defines five stages of the SLA management lifecycle: negotiation and establishment, deployment, measurement and monitoring, corrective management action, and termination. The functionality needed for these various stages is implemented as WSLA services, which are intended to interact across domains. The SLA Compliance Monitor comprises of three WSLA services: SLA Deployment, Measurement, and Condition Evaluation. However, in order to be a generic auditing framework SLA specific elements in the WSLA Framework need to be generalized.

In other more traditional areas of auditing, continuous efforts are done in providing software tools and expert systems to aid human auditors in carrying out the audit. There are approaches to automate certain tasks of financial audit, e.g., through the use of Artificial Intelligence [18], electronic audit data warehouses, and data marts [12]. In this area, audit software tools are utilized to help a human auditor in the analysis of transactional data [1], [2]. However, human interventions are still needed.

3 A Generic Model for Automated Auditing

The different areas of auditing follow different auditing objectives and have different tasks and characteristics, but they share a common model. A general model for automated auditing requires a general definition of auditing. The definitions given by the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) and the Internet Engineering Task Force (IETF) focus on auditing in the area of security [13], [14]. Both define the term Security Audit in a similar way. The U.S. Committee on National Security Systems defines the term Audit instead of Security Audit [17]. Although this definition does not emphasize security, several other auditing areas are not covered. Therefore, a concise, general definition of Audit is given: "An Audit is a systematic and independent examination of facts on system activities to determine the degree of compliance with a pre-defined set of specifications." Auditing is the process of conducting an Audit, and an Auditor is an entity that carries out the Audit.

Fig. 1 depicts a class diagram of the generic auditing model in the UML (Unified Modeling Language) notation. The model consists of 8 classes divided into 3 roles, i.e., Auditor, Auditee, and Accountant, and 5 data, i.e., Audit References, Compliance Specifications, Activities, Facts, and Audit Reports. Compliance Specifications are a set of specifications derived from laws or regulations, contracts or agreements, pre-established policies, and procedures, which the particular system under audit, i.e., the Auditee, has to follow. Note that “follow” means either to meet a specification to achieve an expected state or to avoid meeting a specification to not reach an unexpected state.

An Auditee carries out Activities to achieve a certain goal. The goal itself is irrelevant, however, it is expected that in performing those Activities the Auditee follows the Compliance Specifications. Therefore, those Activities are observed by an Accountant, who records and stores them as Facts. Facts about such Activities reveal whether the targeted Compliance Specifications hold. A Fact is, therefore, a piece of information presented as having an objective reality. A Fact can be accompanied by an Evidence which furnishes a proof, i.e., that ascertains the truth of a matter, thus, increases the informational reliability of a Fact. Evidences are obtained technically through non-repudiation mechanisms, which can be used, e.g., to prove service consumptions [8].

An Auditor conducts an Audit by evaluating Facts based on related Compliance Specifications to detect violations. An Audit has to be conducted according to valid procedures, standards, laws, or regulations, being termed Audit References. Hence, Audit References define the legal or generally accepted way to conduct an Audit in a particular area of auditing. While an Auditee has to follow Compliance Specifications, an Auditor has to follow Audit References. Thus, activities of an Auditor can be subject to the Audit by another Auditor. The Auditor generates an Audit Report based on the result of the Audit. This report can be consulted by the Auditee to avoid further violations.

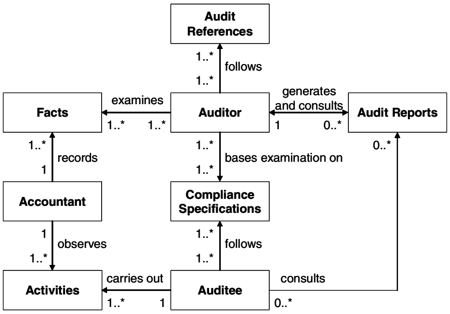


Fig. 1. Generic auditing model

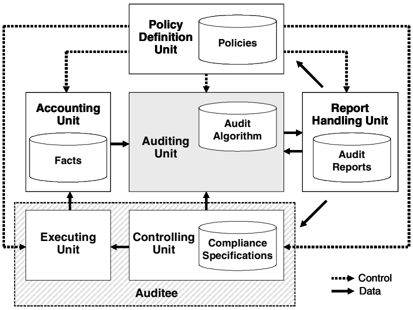


Fig. 2. Generic auditing architecture

4 Architecture

Driven by the general model, the generic auditing architecture is designed, which is applicable to different auditing areas. Architectural components can be distributed across domains, and a suitable approach for an efficient Automated Auditor is included.

4.1 The Design of the Generic Auditing Architecture

Fig. 2 depicts the generic auditing architecture having the Auditing Unit as a major component, which contains a set of Automated Auditors. Automated Auditors may interact with each other, if required, in conducting a particular Audit. The Auditing Unit implements the Audit Algorithm of a particular auditing application. An Audit Algorithm is a technical description of Audit References.

An auditing process requires at least two types of inputs: Compliance Specifications and Facts. Facts describe what actually happens, whereas Compliance Specifications describe expected situations. An Auditee should follow Compliance Specifications in carrying out Activities, hence, an Auditee is divided into a Controlling Unit and an Executing Unit. The Controlling Unit defines Activities to be carried out by the Executing Unit and makes sure that the Compliance Specifications are held. The Controlling Unit provides for Compliance Specifications, whereas the Accounting Unit delivers Facts about Activities performed by the Executing Unit to the Auditing Unit.

The result of an Audit is a report with statements on the degree of compliance of the Auditee's Activities with respect to pre-defined Compliance Specifications. In certain cases, results of previous Audits serve as an input to the current Audit, therefore, the flow of Audit Reports between Auditing Unit and Report Handling Unit is bi-directional. The Report Handling Unit is responsible for maintaining Audit Reports.

The generic auditing architecture uses a policy-based approach to configure and to control the behavior of different units. The policy-based approach offers the advantage of being able to separate decision taking instances from executing instances, hence, it allows for a modular structure. Of course, different modular decision systems may be applied as well. Policies are defined and managed by a Policy Definition Unit. As an example, Audit Policies can be defined to influence the behavior of an Auditing Unit in conducting an Audit without violating Audit References.

In many cases, if the result of an Audit reveals violations to a particular Compliance Specification, a pre-defined corrective action needs to be taken. The action is carried out either by the Report Handling Unit, Policy Definition Unit, or Controlling Unit. Therefore, Audit Reports are also accessible by the Policy Definition Unit and Controlling Unit (cf. Fig. 2). There are three possible causes of a violation: (1) Inappropriately set up Audit Policies, (2) Imprecise Compliance Specifications, and (3) Executing Unit did not perform as expected. In general, cause (1) and (2) should not happen in normal operation, but can happen during learning or experimenting phase. Hence, the generic architecture foresees 3 different feedback control mechanisms to cope with violations:

1. Feedback to the Auditor through changing Audit Policies. This is useful, when violations have occurred unexpectedly due to Audit Policies being inappropriately set up. The Policy Definition Unit must be able to decide whether an Audit Policy is appropriately set up or not, which is a difficult task without human intervention.
2. Feedback to the Auditor through modifications of Compliance Specifications. This is useful, when violations have occurred unexpectedly due to imprecise Compliance Specifications. Here, the Controlling Unit must be able to decide, whether a Compliance Specification is precisely specified or not, which is also a difficult task without human intervention.
3. Feedback to the Executing Unit by the Controlling Unit through reconfiguration. This should be the case if the Executing Unit did not perform as expected.

Obviously, the use of one mechanism does not preclude the use of other mechanisms at the same time, because each mechanism serves a different purpose.

4.2 Distributed Architecture Across Administrative Domains

In some areas, e.g., SLA compliance verification, auditing is applied in a multi-administrative domain (AD) environment. The application of the generic auditing architecture is not restricted to a single AD, instead, architectural components can be distributed across several ADs. However, such a distributed architecture requires trust among ADs.

An AD can offer an accounting or auditing function as a service to other ADs. In Fig. 3 the AD 1 provides an auditing service, whereas the AD 3 offers an accounting service. The AD 2 uses the auditing service of AD 1, who further makes use of the accounting service of the AD 3. Here, AD 4 implements all the functions, but still needs additional accounting information from the AD 3 to conduct the Audit. Note that the Executing Unit and the feedback arrows are not shown to avoid overloading.

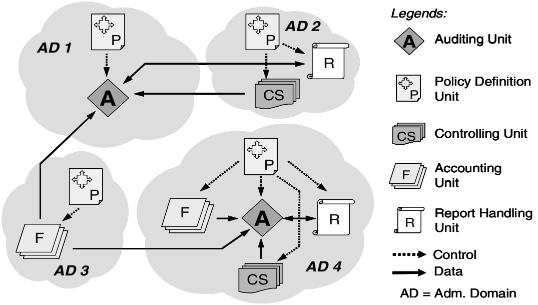


Fig. 3. Multi-administrative domain architecture

4.3 Automated Auditor's Internal Architecture

As described, an Auditing Unit contains a set of Automated Auditors and it implements the Audit Algorithm of a particular auditing application. This means that Automated Auditors have the task to execute the Audit Algorithm. In order to reduce implementation complexity and to achieve modularity, the following assumptions are made in designing the architecture of an Automated Auditor:

- An Auditing Unit deals with a set of Compliance Specifications. Without loss of generality each Automated Auditor is assumed to be responsible for a particular Compliance Specification.
- Each Compliance Specification contains a set of Compliance Conditions linked by a logical expression. Hence, the result of the evaluation of each condition as well as the evaluation of the logical expression linking all the conditions determine the compliance of relevant Facts with a Compliance Specification.
- A common Audit Algorithm exists which is valid for most auditing applications.

The approach developed here proposes the following common Audit Algorithm:

1. Interpret and apply valid Audit Policies during the Audit.
2. Interpret the assigned Compliance Specification, for which the Automated Auditor is responsible.

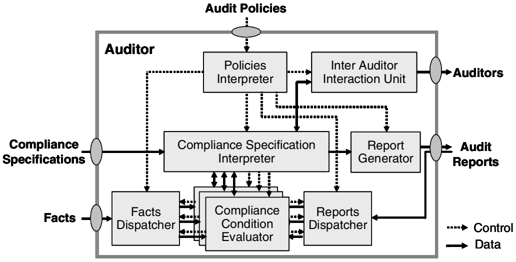


Fig. 4. Automated Auditor's internal architecture

3. Retrieve relevant Facts and Audit Reports based on the Compliance Conditions.
4. Evaluate Facts and Audit Reports whether they meet Compliance Conditions. Evaluate the logical expression linking all Compliance Conditions.
5. Generate a report as a result of the evaluation.

Fig. 4 shows the architecture of the Automated Auditor to execute the proposed Audit Algorithm. The Policies Interpreter (PI) takes policy decisions and configures other components based on Audit Policies. The Compliance Specification Interpreter (CSI) retrieves the Compliance Specification assigned to the Auditor based on the configuration information from PI. It instantiates Compliance Condition Evaluators (CCE) and gives each CCE a Compliance Condition to evaluate. Each CCE subscribes relevant types of Facts and Audit Reports from the Facts Dispatcher (FD) and the Report Dispatcher (RD), respectively.

FD is responsible for retrieving, filtering, and distributing Facts to the respective CCE. A similar function is assigned to RD. Each CCE examines received Facts and relevant Audit Reports in evaluating the condition given by the CSI and sends the result of this evaluation to the CSI. The CSI determines whether there is a violation of the Compliance Specification based on the result of each CCE. The decision of the CSI is sent to the Report Generator which is responsible for composing the Audit Report in a pre-defined format and storing it into a pre-configured location. If interactions among Automated Auditors are required to conduct a particular Audit, interactions are handled by the Inter Auditor Interaction Unit (IAIU). This requirement is stated either in Audit Policies or Compliance Specifications. In both cases the CSI is generally involved.

5 Application of Auditing Model and Architecture

The model developed and the generic architecture designed are applicable to various auditing areas, especially, SLA compliance verification.

5.1 Application of the Model

To start with the verification, Table 1 maps each generic class of the diagram in Fig. to different entities of three different auditing areas: SLA compliance verification, intrusion detection, and financial auditing.

Table 1. Application of the generic auditing model to different auditing areas

Model	SLA Compliance Verification	Intrusion Detection	Financial Auditing
Auditor	SLA Compliance Auditor	Intrusion Detection Engine	Auditor
Auditee	Service provisioning entities	Users of network infrastructures and services	Company, including its Accountants
Accountant	Meter and accounting entities	Sensors (usage monitoring and logging entities)	Company's Accountants
Audit References	Agreed procedures to conduct an Audit	Intrusion detection methods	Auditing standards, *e.g.*, Generally Accepted Auditing Standards (GAAS)
Compliance Specifications	Service Level Agreements	Signature-based: intrusion rules, patterns (signatures) vs. anomaly-based: heuristic rules, normal states or behavior	Accounting standards, *e.g.*, Generally Accepted Accounting Principles (GAAP)
Activities	Service deliveries	Usage of network infrastructures and services	Accounting of business transactions, in particular generation of financial statements
Facts	Meter data or accounting data, and event logs	Network traffic data and event logs	Financial statements
Audit Reports	SLA violation reports	Intrusion reports and alarms	Audit reports

5.2 Application of the Architecture: SLA Compliance Verification

To outline the application of the generic architecture to the Internet, the SLA compliance verification is taken. An SLA contains amongst others: customer ID, subscription start date, subscription end date, subscribed services and service classes, prices, SLOs, remedies in case of SLA violations, and limiting conditions. Information in an SLA is used for access control, meter configuration, and SLA compliance verification.

The following example in Fig. 5 provides a basis for the detailed description on the application of the generic architecture to SLA compliance verification. Provider P is a network operator providing QoS-enabled network services to her customers. The provider's network is connected to the Internet via two Border Routers (BRs), and customers can access the Internet via one of the three Access Routers (ARs). Provider P offers different network service classes to meet different customer and application needs.

The throughput parameter is part of SLAs between Provider P and her customers, and this parameter shall be guaranteed. In order to be fully precise in the definition of this guarantee, downlink and uplink traffic as well as incoming and outgoing traffic need to be distinguished (Fig. 5). The traffic coming from a terminal is the uplink traffic of this terminal and the traffic going to a terminal is the downlink traffic. Traffic entering the network is the incoming traffic, whereas traffic leaving the network is the outgoing traffic. Based on the above description two kinds of throughput guarantee can be defined: downlink and uplink throughput guarantee. Each guarantee is held in an SLO.

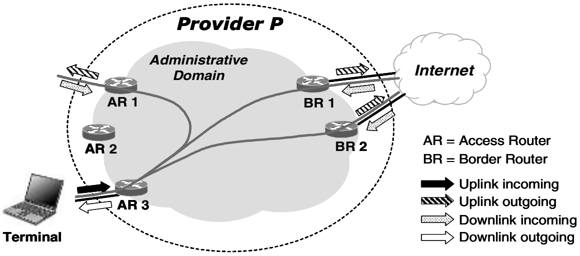


Fig. 5. Different traffic for throughput definition

Downlink throughput guarantee is defined as follows: "The percentage rate reduction in downlink traffic of a network Service Class between the incoming and outgoing traffic will be within dR_d (downlink rate reduction tolerance) in every pre-defined Metering Interval (MI) during the whole session. The rate of the downlink incoming traffic used in the calculation is at most equal to the downlink committed rate Rc_d." This guarantee defines a condition which can be expressed mathematically using the formula:

$$1 - \frac{Ro_d(t)}{Min\,(Rc_d, Ri_d(t))} < dR_d \quad . \tag{1}$$

In (1) $Ro_d(t)$ is the downlink outgoing rate, whereas $Ri_d(t)$ is the downlink incoming rate. Values of Rc_d, dR_d, and *MI* are determined by the service class. Uplink throughput guarantee is defined in a similar way.

Executing Unit. Service provisioning entities are the provider's network infrastructure, in particular QoS-enabled routers and switches. They must deliver services according to SLAs, hence, they determine the Executing Unit.

Controlling Unit and Compliance Specifications. Provider P defines and manages SLAs. She configures and operates service provisioning entities. Therefore, Provider P represents the Controlling Unit. In the area of SLA compliance verification, Compliance Specifications (CS) are derived from concluded SLAs and Compliance Conditions (CC) within a CS are determined by the related SLO. Obviously, an SLA contains more information than needed for defining CSs. Here, it is sufficient if a CS comprises of SLO metrics, service class, and condition expression. As Provider P only provides a single service, namely a network service, it is not required to have the name "network service" as subscribed services in the CS. An example of a CS reads as follows:

Specification Number: 001,
Metrics: Downlink Throughput,
Service Class: A (Rc_d = 1 Mbps, dR_d = 0.05),
Conditions: 1 - Ro_d(t) / Min(Rc_d, Ri_d(t)) < dR_d,

CS 001 is valid only for customers of service class A, but an explicit list of customer IDs in CS 001 is not needed due to the fact that this case does not deal with auditing of access control or intrusion detection, but with auditing of SLA compliance. Access control entities are responsible to ensure that only authorized customers can access services. Hence, recorded Facts are assumed to contain correct/authorized customer IDs.

CS 001 is a simple CS, where parameter Conditions comprise of a single relational expression. Other CSs can be very complex, in which parameter Conditions contain several relational expressions linked by logical operators, and the evaluation is triggered by the occurance of some other Facts.

Accounting Unit and Facts. A meter is deployed in each AR and BR to capture the data rate of customers' traffic. To allow for rate measurements of different service classes, the traffic of each service class must be marked accordingly, e.g., by using Differentiated Service Code Points (DSCP). For the examination of downlink throughput guarantees the following information must be collected by meters: router interface, DSCP, destination IP address, meter interval start time, meter interval end time, and volume. The router interface determines whether the traffic being metered is an outgoing or incoming traffic. The DSCP determines the service class, i.e., the agreed Rc_d and dR_d. The destination IP address represents the customer of the downlink traffic. Time interval and volume are used to calculate the traffic rate.

To ease auditing, related meter data need to be pre-processed by accounting entities. This pre-processing includes data aggregation from different meters and information mapping. In case of downlink throughput guarantee, the resulting accounting record contains the following information: customer ID, service class, meter interval end time, downlink incoming rate, and downlink outgoing rate. Obviously, meter data and accounting data represent Facts about service deliveries and their quality level. Hence, meters and accounting entities build up the Accounting Unit of the generic architecture. To avoid manipulation of meter data or accounting data by provider (or customer) the measurement and accounting task can be carried out by a third party.

Auditing Unit: Automated Auditors. Automated Auditors have the task to evaluate whether the accounting data meet the SLOs specified in SLAs. Auditors deal with a set of CSs and a large number of accounting data recorded during service usage by a large number of customers. Therefore, it is reasonable to run a set of Automated Auditors, where each is responsible for a single CS and a fraction of the accounting data. In the example with Provider P, one way to divide the accounting data is by grouping them based on any combination of SLO metrics, service class, and customer ID. For example, an Automated Auditor AA1 is assigned to evaluate downlink throughput of traffic of service class A addressed to customer with ID ranging from 101 to 200. This means, an Accounting Unit must be able to selectively distribute accounting data to each Auditor.

As already described, CSI determines the core component of an Automated Auditor. The CSI of AA1 interpretes CS 001 and instantiates a single CCE, since parameter Conditions in CS 001 consist only of one relational expression. The CCE configures the FD to retrieve customer ID, downlink incoming rate $Ri_d(t)$, and downlink outgoing rate $Ro_d(t)$ from accounting data with parameters: Metrics = Downlink Throughput, Service Class = A, and Customer ID between 101 and 200. If there is a Fact with $Ri_d(t)$ and $Ro_d(t)$ values which do not satisfy the specified condition, a violation report is created.

Report Handling Unit and Audit Reports. Based on the evaluation result of the Auditing Unit, the Report Handling Unit determines, which SLA is violated and what actions, if any, need to be performed. It also keeps Audit Reports in a pre-defined format for later use. In the example the violated SLA is found by matching the customer ID in the SLA with the customer ID of the violating accounting record.

Policy Definition Unit and Policies. Provider P configures various units and defines policies to be consulted by those units, hence, Provider P represents the Policy Definition Unit. An example for a configuration item is the communication interface of an Automated Auditor. An Automated Auditor communicates with a Controlling Unit, a set of Accounting Units, a set of Report Handling Units, and if required, other Automated Auditors. Which Accounting Units and Report Handling Units have to be contacted depends on which Facts and Reports are to be retrieved and where to store all resulting Reports. Configuration parameters include a URL, i.e., IP address or hostname, port number, and protocol. The following configuration example states that in order to obtain data on downlink throughput of class A traffic for all customers, the Accounting Unit running at host testbed_db.ethz.ch should be contacted via port 13000 using the Diameter protocol.

```
Metrics: Downlink Throughput
Service Class: A
Customer IDs: ALL
Accounting Unit: testbed_db.ethz.ch:13000:diameter
```

Audit Policies are useful to influence the behavior of an Automated Auditor. For example, a policy can be defined for the Automated Auditor AA1 to treat specific accounting records differently. Assume that Customer ID 113 is allowed to send traffic with a downlink committed rate 10% above the rate for service class A during the month April 2005. In this regard an Audit Policy for AA1 reads as follows:

```
if (Customer ID = 113 and Meter Timestamp within April 2005)
  then use Rc_d = 1.1 Mbps.
```

Policies for other units may also be defined, if needed. For example, a Meter Policy for the Accounting Unit can be specified to adapt metering intervals to network load, because the smaller the interval, the more meter data need to be transfered. However, the chosen interval may not lie outside of the range defined in the SLA.

6 Summary, Conclusions and Outlook

Although different auditing areas require different audit tasks, they share a common model, which is shown by developing a generic auditing model and mapping elements of this model to entities in different auditing areas. A generic auditing architecture has been designed based on this model and applied to SLA compliance verification.

The implementation of an auditing framework has been planned and a first effort to develop a Compliance Specification Interpreter is being made. This requires the definition of a general compliance specification language, which is currently being studied. The ultimate goal is to show that it is feasible to provide a generic framework for most areas of automated auditing. In this regard, the model and architecture must be supported with formalizations of various aspects of auditing, including algorithm, policy definition, compliance specification, fact representation, and interface definition between those architectural components. Additionally, the privacy concern in inter-domain applications needs to be solved, and a set of use cases needs to be analysed. These form the main part of further research work.

Concluding, the generic model and architecture provide a common and flexible basis for further development in various auditing areas, in particular security auditing, SLA compliance verification, and business auditing. The availability of an auditing framework based on this generic model and architecture is very important, as automated auditing becomes crucial in many business and security processes, and a fast as well as efficient automated auditing is essential. In turn, future efforts to implement the auditing for specific applications are reduced heavily with the help of this framework.

Acknowledgments

The work has been performed partially in the framework of the EU IST project Daidalos (FP6-2002-IST Contract No. 506997), where ETH Zürich has been funded by the Swiss Bundesministerium für Bildung und Wissenschaft (grant No. 03.0141). The authors would like to thank their project partners, especially Portugal Telecom Inovação and Fraunhofer-Gesellschaft FOKUS.

References

1. ACL Services Ltd.: ACL Tops 2004 Internal Auditor Software Survey. (2004)
2. CaseWare IDEA Inc.: IDEA: Product Profile. (2004)
3. Daidalos: A4C Framework Design Specification. Deliverable D341 (2004)
4. D'Antonio, S., Esposito, M., Gargiulo, M., Romano, S.P., Ventre, G.: A Component-based Approach to SLA Monitoring in Premium IP Networks. First Intl. Workshop on Inter-Domain Performance and Simulation, Salzburg (2003)
5. Denning, D. E.: An Intrusion-Detection Model. IEEE Transactions on Software Engineering, Vol. SE-13, No.2 (1987) 222-232
6. G-NE GmbH: Konzeptionsansatz: Qualitätssicherung in IT-Outsourcing-Projekten mittels einer unabhängigen Prüfinstanz. Confidential Document (2002)
7. Hasan, Stiller, B.: Auditing Architecture for SLA Violation Detection in QoS-Supporting Mobile Internet. IST Mobile and Wireless Comm. Summit, Vol. 1. Aveiro, Portugal (2003)
8. Hasan, Stiller, B.: Non-repudiation of Consumption of Mobile Internet Services with Privacy Support. IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communications (to be published), Montreal, Canada (2005)
9. Itellix Software: Wisiba: Datasheet. (2003)
10. Keller, A., Ludwig, H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Journal of Network and Systems Management, Vol. 11, Issue 1 (2003) 57 - 81
11. Lundin, E., Jonsson, E.: Survey of Intrusion Detection Research. Technical Report 02-04, Department of Computer Engineering, Chalmers Univ. of Technology, Göteborg (2002)
12. Rezaee, Z., et. al.: Continuous Auditing: Building Automated Auditing Capability. Auditing: A Journal of Practice & Theory, Vol. 21 Issue 1 (2002) 147-163
13. Shirey, R.: Internet Security Glossary. IETF, RFC 2828 (2000)
14. Study Group on Communication Systems Security: Compendium of approved ITU-T Security Definitions. (2003)
15. Softek Storage Solutions Corporation: SOFTEK EnView: Datasheet. (2004)
16. Telemanagement Forum: SLA Management Handbook, V1.5. GB917 (2001)
17. U.S. Committee on National Security Systems: National Information Assurance Glossary. (2003)
18. Vasarhelyi, M.A.: Artificial Intelligence in Accounting and Auditing, Volume IV: Towards New Paradigms. (1997)

Utilization and SLO-Based Control for Dynamic Sizing of Resource Partitions

Zhikui Wang, Xiaoyun Zhu, and Sharad Singhal

Hewlett Packard Laboratories,
1501 Page Mill Rd, Palo Alto, CA 94304
{zhikui.wang, xiaoyun.zhu, sharad.singhal}@hp.com

Abstract. This paper deals with a shared server environment where the server is divided into a number of resource partitions and used to host multiple applications at the same time. In a case study where the HP-UX Process Resource Manager is taken as the server partitioning technology, we investigate the technical challenges in performing automated sizing of a resource partition using a feedback control approach, where the CPU entitlement for the partition is dynamically tuned to regulate output metrics such as the CPU utilization or SLO-based application performance metric. We identify the nonlinear and bimodal properties of the models across different operating regions, and discuss their implications for the design of the control loops. To deal with these challenges, we then propose two adaptive controllers for tracking the target utilization and target response time respectively. We evaluate the performance of the closed-loop systems while varying certain operating conditions. We demonstrate that better performance and robustness can be achieved with these controllers compared with other controllers or our prior solution.

1 Introduction

Resource partitioning is a type of virtualization technology that enables multiple applications to share the system resources on a single server while maintaining performance isolation and differentiation among them [1][2][3]. On most current systems, partition sizes are pre-determined and allocated to applications by system administrators, posing a challenging configuration problem. On the one hand, each partition has to be provided with enough resources to meet service level objectives (SLOs) of the applications hosted within it in spite of changes in workloads and the underlying system. On the other hand, excessive over-provisioning makes inefficient use of resources on the system. Offline capacity planning or calendar-based scheduling using profiles of past application resource usage are not always accurate or up-to-date and cannot handle unexpected short-term spikes in demand.

Our work aims to develop formal control-theory based techniques to automatically size a resource partition based on its CPU utilization, the SLO and the time-varying workload of its hosted applications. This work is the continuation of our earlier work in [4] where we used a resource partition to host an Apache Web server and designed and implemented an adaptive PI controller to regulate the mean response time (MRT) of HTTP requests around a target value. The controller self-tunes its gain parameters

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 133 – 144, 2005.

based on online estimation of the dynamic model. In this paper, we describe a new set of modeling experiments and demonstrate how the system's input-output relation changes with various operating conditions of the system. As a result, we show that controlling the MRT using the CPU entitlement alone is effective when the Web server partition's CPU utilization is close to its CPU entitlement, but may not work well when the application is underutilizing its entitled CPU. We present an alternative design for controlling the relative utilization of the partition, scalable to the time-varying workload. We also propose to incorporate CPU utilization information into the control of SLO-based metrics so that the closed-loop system achieves more robust performance across different operating regions.

This paper is organized as follows. In section 2, we describe the technology, the overall architecture and the test bed in our case study. Related work is reviewed in Section 3. Section 4 describes how the input-output behavior of the system was modeled, and discusses its implication on control designs. Section 5 presents different controllers and their performance evaluation using our test bed. Finally, we summarize our results, along with directions for future work in Section 6.

2 A Case Study Using a Feedback Control Approach

We conducted the case study where we used the *HP-UX Process Resource Manager* (PRM) [1] as an example of the resource partitioning technology. PRM is a resource management tool that can be used to partition a server into multiple PRM groups, where each PRM group is a collection of users and applications that are joined together and allocated certain amounts of system resources, such as CPU, memory, and disk bandwidth. If CPU or memory capping is enabled, PRM ensures that each PRM group's usage of CPU or memory does not exceed the cap regardless of whether the system is fully utilized. We consider an FSS (Fair Share Scheduler) PRM group that is assigned a percentage of the CPU cycles (referred to as "CPU entitlement") by specifying a number of shares. Because this percentage is enforced by the scheduler in the HP-UX kernel, it can be changed at any time thereby enabling dynamic sizing of the PRM group. We describe the architecture and test bed setup in this section.

Figure 1 illustrates a shared server that has *m* resource partitions, where each partition can be a PRM group. We consider the scenario where each PRM group is used to host one application. The resource controller interacts with each partition *i* through two modules, A_i and S_i, where S_i is the sensor that periodically measures the performance metric for application *i*, and A_i is the actuator that dynamically sets the CPU entitlement for partition *i* according to the output of the resource controller. The timing of the controller is based on the notion of a "sampling interval". At the beginning of each sampling interval, the controller collects from S_i the measured performance for the last sampling interval, compares it to the desired performance, computes the needed CPU entitlement for the current sampling interval using certain control algorithms and passes it to A_i for actuation. In the remainder of this paper, we focus on resource control for one such partition. The results should be extensible to controlling multiple partitions with multiple resource types using any partitioning technology.

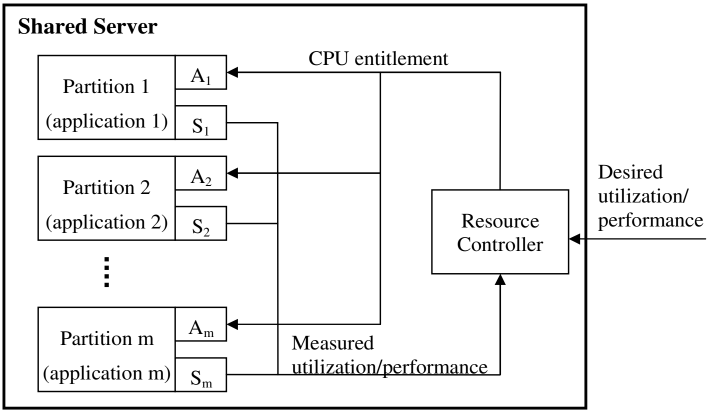


Fig. 1. CPU entitlement control system architecture

In the case study, we took the *Apache Web server* (version 2.0.52) as an example of the hosted applications. We set up an FSS PRM group on an HP-UX server to host the Web server. We refer to this PRM group as the "Web server partition". We used a modified version of *httperf 0.8* (ftp://ftp.hpl.hp.com/pub/httperf) on a Linux 2.4.18-3 client to continuously send HTTP requests to the Web server and to log the response time of every request. We developed a sensor module that parses the httperf log and computes the mean response time (MRT) of all the requests completed during each sampling interval. We also used a PRM provided utility *prmmonitor* to measure the average CPU utilization of a partition for every interval. The CPU entitlement (with capping enabled) for the Web server partition can be adjusted at the beginning of every interval to bound the percentage of CPU cycles used by the Web server in that interval. We chose the simplest possible workload where a single static page was repeatedly fetched from the Web server at a fixed rate, ensuring that CPU was the only potential bottleneck resource in the system as the workload intensity varied.

3 Related Work

Our approach differs from prior work on operating systems support for server resource reservation and enforcement [5]-[7] or scheduling [8][9] in that it is more generic and can be used on any commodity operating system that supports a resource partitioning technology, and applications that can be hosted inside a partition. In [10] a feedback-driven adaptive scheduler was presented to allocate a percentage of CPU cycles to a thread. In contrast, our controller allocates a percentage of CPU cycles to a whole application so that the assigned CPU entitlement can be tied directly to the application's SLO. Although the proposed feedback loop is already in use in some existing workload management tools [11][12], our approach is distinct in that we rely on classical control theory to guide the design of the algorithms.

Feedback control theory has been applied to solve a number of performance or quality of service (QoS) problems in computer systems in recent years. (See [13][14] and the references therein.) The effect of this approach depends heavily on the fitness of the mathematical models used to characterize the dynamic behavior of the systems. Much prior work employs a "black-box" approach and uses linear input-output models to capture the dynamic relation between control knobs (inputs) and performance metrics (outputs). However, a single linear model is often insufficient to uniformly capture a system's behavior under all operating conditions. More recent work addresses this issue by applying adaptive control theory to computer systems such as storage systems [15], resource containers [4] and caching services [16]. This approach allows the parameters of the linear models to automatically adapt to changes in operating conditions using online system identification. In [17] the authors offered insights into how to obtain appropriate models for the actuator, sensor, and the controlled system using benchmarking and linear regression based estimation techniques, while using CPU utilization as the output. The resulting relation between the adaptation level and the CPU utilization is a time-varying static gain with no dynamics but with a time delay. In this paper, we focus on the system's nonlinear and bimodal behavior, and present quantitative study on model fitness and variability.

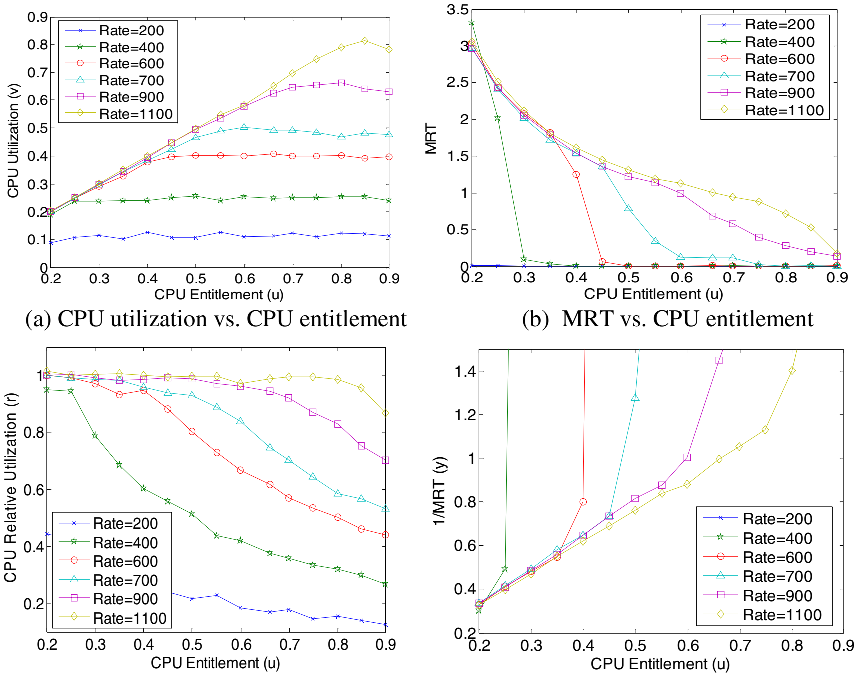
Performance control of Web servers has been studied extensively in the literature. For instance, application-level mechanisms were proposed in [18][19][20] to provide different levels of service to requests of different classes. While these approaches were mainly based on heuristics or queuing models, other work has applied classical control theory to manage Web server delay or server resource utilization using admission control [21] and content adaptation [17], connection scheduling and process reallocation [22], or application parameter tuning [23]. All of these methods require modification to the server application software (with the exception of [20]), which may not be feasible for other enterprise applications. Our focus is not controlling Web server performance in particular, but rather providing a general approach for dynamic sizing of any resource partitions.

4 Modeling of the Input-Output Relation

We first describe a set of modeling experiments and demonstrate the nonlinear and bimodal behavior of the system.

4.1 Static Input-Output Relation

To understand the system's long-term average behavior in the whole operating range, we varied the CPU entitlement (denoted by u) for the Web server partition from 0.2 to 0.9, at 0.05 increments. At each setting, the Web server was loaded for 60 seconds with a fixed workload, while the average CPU utilization (denoted by v) of the Web server partition was observed and the MRT of all requests returned during this period was computed. Figure 2 shows the static relation between the CPU entitlement, the (absolute and relative) CPU utilization, and the MRT for different workload intensities ranging from 200 to 1100 requests/second (or *r/s*). Note that each data point is the average of 10 samples obtained from 10 repeated experiments. In addition to u and v, let y denote the inverse of MRT (1/MRT), and r denote the relative CPU utilization of the partition, i.e., $r = v / u$.



(a) CPU utilization vs. CPU entitlement

(b) MRT vs. CPU entitlement

(c) Relative CPU utilization vs. CPU entitlement

(d) 1/MRT vs. CPU entitlement

Fig. 2. Long-term relation between CPU entitlement, CPU utilization and MRT

Our key observations from these figures follow:

- As shown in Figure 2(a), for any given request rate, as the CPU entitlement varies, the CPU utilization demonstrates a clear *bimodal* behavior that can be approximated using the following equation:

$$v = \begin{cases} u, & \text{if } u < V, \\ v, & \text{if } u >= V. \end{cases} \tag{1}$$

  Here V is the maximum portion of CPU needed for a given workload. Figure 2(c) shows a different visualization of the same behavior through the relative CPU utilization. The following equation is equivalent to (1), except expressing the CPU utilization in a relative term:

$$r = \begin{cases} 1, & \text{if } u < V, \\ V/u, & \text{if } u >= V. \end{cases} \tag{2}$$

- Similarly, the same bimodal behavior is observed in the relation between the MRT and the CPU entitlement in Figure 2(b). Since the MRT is clearly a nonlinear function of the CPU entitlement, we plot 1/MRT vs. CPU entitlement in Figure 2(d) to better illustrate the relation. As we can see, when the system is overloaded

(r = 1 in Figure 2(c)), there exists a linear mapping from the CPU entitlement to 1/MRT, and its slope is independent of the request rate. However, when the system is underloaded (r < 1 in Figure 2(c)), 1/MRT increases rapidly with increasing CPU entitlement, indicating a sharp drop in the MRT.

The linear mapping between the CPU entitlement and 1/MRT for the overload region implies that a linear input-output model is plausible for this region if 1/MRT is chosen as the system output. When the system is reasonably underloaded ($r < 0.8$), the MRT becomes independent of the CPU entitlement setting. Therefore, we expect that the MRT is uncontrollable using the CPU entitlement in this region. In the next Section, we verify this behavior using model identification.

4.2 Dynamic Linear Model Identification

We chose the following linear auto-regressive model as the potential one to represent the dynamic relation between the CPU entitlement and the inverse of MRT:

$$y(k) = \sum_{i=1}^{n} a_i y(k-i) + \sum_{j=0}^{m-1} b_j u(k-d-j) + \varepsilon(k), \tag{3}$$

where the parameters a_i, b_j, the orders m, n, and the delay d characterize the dynamic behavior of the system, $y(k)$ is the inverse of MRT for sampling interval k, $u(k)$ is the CPU entitlement for sampling interval k, and $\varepsilon(k)$ is the residual term. For convenience, we refer to such a model as "ARX*mnd*" in the following discussion.

In the experiments, the CPU entitlement was randomly varied in [0.2, 0.8]. The sampling interval was fixed at 15 seconds while the rate varied from 200 *r/s* to 1100 *r/s*. The experiment was repeated for each rate. The model in (3) was estimated offline using least-squares based methods [24] in the *Matlab System ID Toolbox* [25] to fit the input-output data collected from the experiments. The models are evaluated using the r^2 metric defined in *Matlab* as a goodness-of-fit measure. In general, the r^2 value indicates the percentage of variation in the output captured by the model.

Table 1. r^2 values (in percentage) of first-order models

(a) under different workloads

Model	Rate (r/s)					
	200	400	600	700	900	1100
ARX110	-10.2	12.8	2.8	63.1	70.3	78.3
ARX111	-1	6.7	2.7	-5	0.09	6.4

(b) for different input-output pairs

Range of Entitlement	[0.2, 0.5]	[0.5, 0.8]
Ent --> 1/MRT	77.6	26
Util --> 1/MRT	84.3	65.5
Ent --> Util	86	31.5

From the data in Tables 1(a), we can find that a simple linear model does not fit the input-output data when the system is significantly underloaded, i.e., with a rate below or equal to 600 *r/s*. This is consistent with our earlier observation from Figure 2(d). In contrast, when the request rate is above 600 r/s, the ARX 110 model fits quite well, providing a good basis for controller design. Moreover, ARX111 (first-order model with one-step delay) does not explain the system behavior for any request rate, showing that no significant delay is observed in the system dynamics for a sampling interval of 15 seconds. Other observation can be made on time-varying parameters along with change of workload, different model delays when the sampling interval was changed significantly. For more detailed analysis, see [26].

Similar experiments and analysis were repeated for a different server, and the same qualitative results were observed. Our main conclusion is that, due to the existence of first-order ARX models for the dynamic relation between the CPU entitlement and 1/MRT when the system is overloaded, the MRT should be controllable using simple controllers such as the adaptive PI controller used in [4]. On the other hand, it will be quite challenging to regulate the MRT in the underload region because our observations from the modeling exercise suggest that the MRT is simply uncontrollable using the CPU entitlement as the only input.

From Figure 2, we know that the MRT is not correlated with the CPU entitlement in the underload region. However, the MRT should always be dependent upon the real CPU utilization of the Web server process. This was confirmed from the following exercise, where offline identification experiments were repeated when the CPU entitlement was randomly varied in the two regions, as shown in Table 1(b), under a fixed workload of 900r/s. In the underload case where the entitlement range is [0.5, 0.8], 1/MRT is only weakly correlated with the CPU entitlement with r^2=26%. However, the r^2 value of the models between the CPU utilization and 1/MRT is always much higher. Therefore, it should be helpful to introduce the CPU utilization into the control loop for the MRT so that more robust performance can be achieved.

5 Controller Design and Performance Evaluation

The CPU utilization of a Web server is a common metric that is monitored to determine whether more or less CPU resource should be allocated to the server. Compared to SLO-based metrics such as response times, the relative utilization of the resource partition is easier to measure on the server side, more directly related to the CPU entitlement and its control is more intuitive. The downside is that the relation between a given relative utilization level and the client-perceived service level varies with the resource demand of the workload. No guarantees can be given to metrics such as the MRT for an arbitrary workload when only the relative utilization is being controlled. This is in contrast to using the MRT as the controlled output that is more directly related to the SLO but its relation with the CPU entitlement is rather complex. In this section, we present controller designs for dynamic sizing of the Web server partition using both output metrics, and discuss possible ways to combine these two metrics to provide more effective control across the whole operating region.

5.1 Control of Relative Utilization

We first consider dynamic sizing of the Web server partition using its relative utilization, $r(k)$, as the output and the CPU entitlement, $u(k)$, as the input. The goal is to maintain dynamically the relative utilization at a reference value, $r_{ref.}$ This value can be chosen higher for more predictable workloads, and lower for more variable workloads. From offline identification experiments, we observed that $r(k)$ responds quickly to changes in $u(k)$ with negligible delay and inertia when the sampling interval is set at 15 seconds. Therefore, the nonlinear static model (2) can be used to represent the input-output relation.

Define the tracking error at sampling interval *k* as

$$e(k) = r_{ref} - r(k). \tag{4}$$

We can then use the classical integral (I) controller,

$$u(k) = u(k-1) - K_i e(k-1)\,, \tag{5}$$

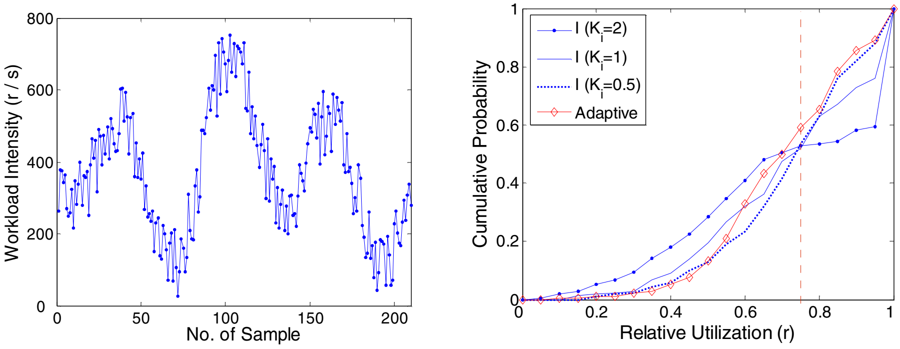
to dynamically tune the CPU entitlement based on the tracking error. In theory, integral control ensures zero steady state error, i.e., the measured relative utilization should converge to r_{ref}, and the integral gain K_i determines the aggressiveness of the tuning. The main challenge here is to choose the right gain parameter such that the closed-loop system is stable, and the relative utilization tracks the reference value as quickly as possible. Although an optimum K_i value may be chosen carefully for certain workload, it may not be applicable to a different workload. Based on the analysis in Section 4, we propose an I controller:

$$u(k) = u(k-1) - K_i(k)e(k-1) \tag{6}$$

with a time-varying adaptive gain:

$$K_i(k) = \begin{cases} \lambda_1 u(k-1)/r_{ref}, & \text{when } u(k-1) = v(k-1), \\ \lambda_2 v(k-1)/r_{ref}, & \text{when } u(k-1) > v(k-1). \end{cases} \tag{7}$$

The intuition behind this adaptive gain is from the bimodal property of the relative utilization w.r.t. CPU entitlement. When the system is overloaded, the CPU utilization is actually capped by the entitlement. The system needs to react fast to the increase of the workload. When underloaded, it is desirable for the system to be conservative to avoid going into the overload region, which may lead to unavailability and large response times. The controller (6-7) can act as expected with different gains in the two regions. It is scalable and adaptive to the workload, and shows good convergence performance when λ_2 is in (0, 2). More analysis can be found in [26].



(a) Time-varying workload (b) CDFs of relative utilization

Fig. 3. Performance of controllers from entitlement to relative utilization

The performance of the adaptive controller (6-7) was tested in an experiment where a synthetic workload as shown in Figure 3(a) was applied. For comparison, our adaptive controller was used along with an I controller with different fixed gains to regulate the relative utilization at around 75%. The target (relative utilization) tracking performance for different controllers is shown in Figure 3(b) by the cumulative distributions of the resulting relative utilization, where the vertical dashed line indicates the ideal distribution. Other performance measures such as average CPU entitlement, throughput, and mean and 95th percentile of response times are compared in Table 2. It can be observed that the adaptive controller achieves better tracking performance upon change of the workload, lower CPU consumption, higher throughput and smaller response times compared to the I controller with fixed gains.

Table 2. Average performance of the utilization controllers

Controller	CPU Ent	Throughput (r/s)	MRT (sec)	95-p RT (sec)
I (Ki=2)	0.43	359	1.02	4.15
I (Ki=1)	0.38	368	0.51	2.49
I (Ki=0.5)	0.38	363	0.64	2.71
Adaptive	0.37	370	0.31	1.69

5.2 Control of Mean Response Time

In this section, we highlight the challenges in controlling the mean response time (MRT). Using examples, we show that even the adaptive PI controller presented in [4] may not work well when a sudden change in the workload pushes the system into the underload region. We then describe a new controller design by introducing the CPU utilization measurement into the control loop.

Based on the analysis in Section 4, we consider an ARX110 model to represent the dynamic relation between the CPU entitlement (u) and the inverse of MRT (y), which is estimated online as done in [4]. Define the tracking error

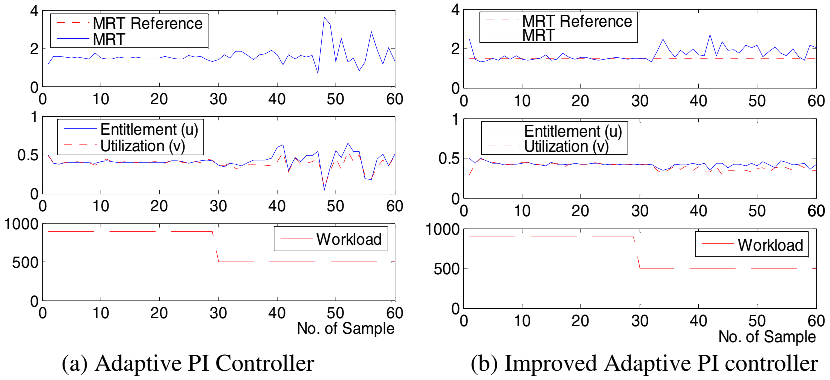
$$e(k) = y_{ref} - y(k), \tag{8}$$

where $y_{ref}(k)$ is the target value for 1/MRT. Then a PI controller implements the following algorithm:

$$u(k) = u(k-1) + (K_p + K_i)e(k-1) - K_p e(k-2) \tag{9}$$

The closed-loop system is of second order and the gain parameters, K_p and K_i, can be chosen using the pole placement algorithm according to design specifications such as overshoot, rising time and settling time [28].

The controller (9) with adapted parameters was tested in an experiment where the target MRT was fixed at 1.5 seconds, but the rate of the workload was changed from 900 r/s to 500 r/s at the 30th sampling interval, which pushes the system suddenly into the underload region. Figure 4(a) shows the performance of the closed-loop system, where we can see that both the CPU entitlement and the resulting MRT became unstable because the loss of controllability of the MRT by the CPU entitlement leads



(a) Adaptive PI Controller (b) Improved Adaptive PI controller

Fig. 4. Performance of controllers from CPU entitlement to MRT

to over-provisioning of the CPU resource. This is consistent with our observation from Figure 3(b) that the MRT is not a stable metric in the underload region.

Given the suggestion of Figure 3(a) that the CPU utilization is a more stable metric, we propose one design that attempts to incorporate the measured CPU utilization into the control loop to extend the controllable region, as illustrated in Figure 5, where G_2 is the mapping from CPU entitlement to CPU utilization, and G_1 is the mapping from CPU utilization to 1/MRT.

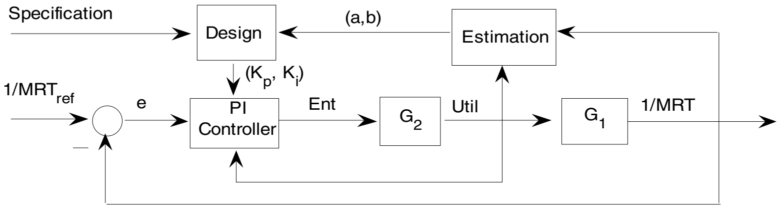


Fig. 5. An adaptive control loop with incorporation of measured CPU utilization

From Table 1(b) in Section 4.2, we know that the CPU utilization has a tighter relation with the MRT than the CPU entitlement does in the underload region. In the following design, the ARX110 model was estimated online between the measured CPU utilization (*v(k),* as the input) and 1/MRT (*y(k),* as the output). Moreover, the term *u(k-1)* in the PI controller in (9) is replaced by *v(k-1)* as follows:

$$u(k) = v(k-1) + (K_p + K_i)e(k-1) - K_p e(k-2) \tag{10}$$

The parameters were chosen according to the same specification as in the prior designs. The previous experiments with varying workload intensity were repeated using the new controller in (10). The closed-loop performance is shown in Figure 4(b), which shows that the stability of the system is maintained, even with a significantly reduced workload. In this control design, introducing the CPU utilization into the model estimation leads to a more truthful and stable model. Moreover, over-tuning of the CPU entitlement can be avoided since it is based on the measured utilization. However, one implicit assumption in this solution is that the utilization measurement tracks the entitlement immediately, that is, $G_2=1$. This is satisfied in the overload region where $v(k)=u(k)$. Offset exists in the underload region between the expected value of the utilization and its measurement. That is why, as shown in Figure 4(b), the measured MRT is above the target value when the system is underloaded. This steady-state error can be estimated approximately and fixed partially as suggested in [26]. Therefore, the proposed solution can improve the robustness of the controller (9) significantly only in or close to the overload region.

6 Conclusions

This paper identifies challenges in applying control theory to dynamic sizing of a resource partition using CPU entitlement as the input and the mean response time or the relative CPU utilization as the output. We recognize that this input-output relation varies significantly as the resource partition moves between the overload and the underload regions, which has a noticeable impact on the performance of any controller design. We evaluate the closed-loop performance of an adaptive integral controller for controlling relative utilization of a resource partition. We also present a new adaptive controller design for regulating the mean response time that incorporates information on measured CPU utilization and improves the robustness of prior adaptive algorithms.

To make the system work well across all operating regions, we need to respect the bimodal behavior of the system and develop a better way to integrate the control of relative utilization (using controller (6-7)) and the response time (using controller (10)) in possibly different regions. This is one topic of our ongoing work. Another interesting direction is to apply the same approach to dynamic sizing of a resource partition in terms of its physical memory allocation. The distinct interaction between application performance and its memory may make it much more challenging to design a sensible controller that works under all operating conditions.

References

[1] HP Process Resource Manager, http://h30081.www3.hp.com/products/prm/index.html

[2] IBM Application Workload Manager, http://www.ibm.com/servers/eserver/xseries/systems_management/director_4/awm.html

[3] SUN Solaris Resource Manager, http://www.sun.com/software/resourcemgr/index.html

[4] X. Liu, X. Zhu, S. Singhal, and M. Arlitt, "Adaptive entitlement control of resource partitions on shared servers," *9th International Symposium on Integrated Network Management,* May, 2005.

[5] G. Banga, P. Druschel, and J.C. Mogul, "Resource Containers: A new facility for resource management in server systems," 3rd USENIX Symposium on Operating Systems Design and Implementation, Feb. 1999.

[6] M.B. Jones, D. Rosu, and M.-C. Rosu, "CPU reservations and time constraints: Efficient, predictable scheduling of independent activities," 16th ACM Symposium on Operating Systems Principles, 1997.

[7] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa, “Resource Kernels: A resource-centric approach to real-time and multimedia systems,” ACM Conference on Multimedia Computing and Networking, 1998.

[8] P. Goyal, X. Guo, and H. Vin, “A hierarchical CPU scheduler for multimedia operating systems,” 2nd USENIX Symposium on Operating System Design and Implementation, October, 1996.

[9] C. Waldspurger and W. Weihl, “Lottery Scheduling: Flexible proportional-share resource management,” 1st USENIX Symposium on Operating System Design and Implementation, 1994.

[10] D.C. Steere, et al., “A feedback-driven proportion allocator for real-rate scheduling,” 3rd USENIX Symposium on Operating System Design and Implementation, 1999.

[11] HP-UX Workload Manager, http://h30081.www3.hp.com/products/wlm/index.html

[12] IBM Enterprise Workload Manager, http://www.ibm.com/developerworks/autonomic/ewlm/

[13] J.L. Hellerstein, Y. Diao, S. Parekh, and D. Tilbury, Feedback Control of Computing Systems, Wiley-Interscience, 2004.

[14] T.F. Abdelzaher, Y. Lu, R. Zhang, and D. Henriksson, ``Practical application of control theory to Web services," invited paper, American Control Conference, June 2004.

[15] M. Karlsson, C. Karamanolis, and X. Zhu, “Triage: Performance isolation and differentiation for storage systems,” 12th IEEE International Workshop on Quality of Service, 2004.

[16] C. Lu, T.F. Abdelzaher, J. Stankovic, and S. Son, “A feedback control approach for guaranteeing relative delays in Web servers,” IEEE Real-Time Technology and Applications Symposium, 2001.

[17] T.F. Abdelzaher, K.G. Shin, and N. Bhatti, “Performance guarantees for Web server end-systems: A control-theoretical approach,” IEEE Transactions on Parallel and Distributed Systems, vol. 13, 2002.

[18] J. Almeida, M. Dabu, A. Manikutty and P. Cao (1998), “Providing differentiated levels of service in Web content hosting,” SIGMETRICS Workshop on Internet Server Performance, June 1998.

[19] L. Eggert and J. Heidemann, “Application-Level differentiated services for Web servers,” World Wide Web Journal, Vol. 3, No. 1, pp. 133-142, March, 1999.

[20] V. Kanodia and E. Knightly, “Multi-Class latency-bounded Web services,” 8th IEEE International Workshop on Quality of Service, June, 2000.

[21] P. Bhoj, S Ramanathan, and S. Singhal, “Web2K: Bringing QoS to Web servers,” HP Labs Technical Report, HPL-2000-61, May 2000.

[22] Y. Lu, C. Lu, T. Abdelzaher, and G. Tao, “An adaptive control framework for QoS guarantees and its application to differentiated caching services,” IEEE International Workshop on Quality of Service, May, 2002.

[23] Y. Diao, N. Gandhi, J.L. Hellerstein, S. Parekh, and D.M. Tilbury, “MIMO control of an Apache Web server: Modeling and controller design,” American Control Conference, 2002.

[24] L. Ljung, System Identification: Theory for the User (2nd Edition), Prentice Hall, 1999.

[25] Matlab System Identification Toolbox, http://www.mathworks.com/products/sysid/

[26] Z. Wang, X. Zhu, S. Singhal, “Utilization and SLO-Based Control for Dynamic Sizing of Resource Partitions”, HP Labs Technical Report, HPL-2005-126, July 2005.

[27] Apache Web server, http://www.apache.org/

[28] K. Astrom and T. Hagglund, *PID Controllers: Theory, Design, and Tuning (2nd Edition),* Instrument Society of America, 1995.

A Decentralized Traffic Management Approach for Ambient Networks Environments

María Ángeles Callejo-Rodríguez, Jorge Andrés-Colás, Gerardo García-de-Blas,
Francisco Javier Ramón-Salguero, and José Enríquez-Gabeiras

Telefónica I+D,
Advanced Networks Planning Department,
Emilio Vargas 6, 28043 Madrid, Spain
{macr327, jorgeac, ggdb, fjrs, jeg}@tid.es

Abstract. This paper presents a decentralized traffic management solution suitable for Ambient Networks environments, where heterogeneous networks will have to cooperate with a high degree of dynamicity, both in traffic patterns and network topologies. Considering IP as the base inter-network technology in these environments, the proposed mechanism autonomously interacts with existing intra-domain routing protocols to improve traffic performance. The proposal has been evaluated by simulation and has been shown how it significantly improves the traffic performance with respect to the solutions currently deployed in networks. For the two simulated scenarios, the proposed solution is able to manage 38% and 15% more traffic than current solutions when the network starts to be congested. Anyway, the behavior of the proposed solution is currently being analyzed in more dynamic scenarios in order to check its goodness for different Ambient Networks environments.

1 Introduction

The Ambient Networks concept [1] aims to provide open and scalable solutions for the near-future networking world where heterogeneous networks, from personal and vehicular networks to access and core transport networks, will have to cooperate to offer ubiquitous communication services to the end-users.

In addition, these scenarios include a wide range of traffic patterns to be carried, with different mobility degrees and performance requirements. Considering IP as the base inter-network technology for Ambient Networks, this paper proposes a decentralized traffic management solution based on the extension of existing static IP intra-domain routing protocols to automatically adapt their routing tables to current traffic dynamics. In this way, traffic management mechanisms autonomously interact with the control plane of the network.

It has to be noted that the work described in this paper focuses on an intra-domain scope. Inter-domain solutions implying routing information exchange among different operators are left for further study.

Regarding intra-domain IP routing algorithms, traditional proposals are based on the dissemination of the network topology in order to allow each router

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 145–156, 2005.

in the network to infer the path with the minimum associated cost. Thus, new routing algorithms are required in order to achieve traffic flows to be forwarded through the available network resources in such a way that no link becomes overloaded and congestion is avoided. For this objective, the usage of routing algorithms based on multipath schemes is required, although their use entails the usage of sub-optimal paths, that is, those paths with costs higher than the optimal ones. This situation can generate routing loops decreasing the efficiency of the routing mechanism. Moreover, routing loops make worse the traffic performance in those scenarios where multipath routing proposals are most interesting: networks with high traffic load, near to or already in a congestion state. Therefore, a thorough study is needed to avoid these loops in the most suitable way. In this paper, a new mechanism for the avoidance of routing loops, called LAP (Loop Avoidance Protocol), is presented. LAP can be used as an extension of any intra-domain IP multipath routing mechanism.

It has to be noticed that solutions based on packet tunneling, such as MPLS-TE (MultiProtocol Label Switching Traffic Engineering) [2] have been discarded beforehand, since they are considered as not flexible enough for the high-dynamic environments envisaged for Ambient Networks. Besides, an IP native solution for traffic management benefits from the scalability and simplicity of IP, which are strongly reduced with the usage of tunneling solutions.

The rest of the paper is structured as follows. In Section 2, a state of the art of routing alternatives for Ambient Networks are presented and the reasoning behind the selection of the MRDV (Multipath Routing with Dynamic Variance) mechanism [3] as the most suitable approach is introduced. Then, a detailed description of LAP is shown in Section 3. Next, Section 4 presents simulation results showing the performance of LAP jointly with MRDV. Finally, Section 5 includes conclusions and further steps.

The research work presented in this paper has been developed within the framework of the Ambient Networks project, partially funded by the European Commission under the Information Society Technology (IST) priority within the Sixth Framework Programme.

2 Routing Alternatives for Ambient Networks

This section surveys existing intra-domain IP multipath routing solutions that can be used in order to optimize network resources in an Ambient Networks environment.

The most deployed multipath routing algorithm in current IP networks is ECMP (Equal-Cost MultiPath) [4], which is inherently supported by common intra-domain routing protocols, such as OSPF (Open Short Path First) [5] and ISIS (Intermediate System to Intermediate System) [6]. In ECMP, all paths with minimal cost are equally used to route traffic. Nevertheless, its scheme does not split the traffic according to a balanced load criterion, as all paths are required to have the minimal cost.

As a more dynamic approach, OMP (Optimized Multi-Path) [7] allows routers to shift load from heavily loaded paths to less loaded ones by means of the use of the global state-network information: new paths can be inferred by other routers in the network since updated and accurate information about the link loads of all the nodes in the network must be exchanged; and thus make OMP not scalable enough in those scenarios where traffic demands are highly variable. Another algorithm, AMP (Adaptive MultiPath) [8] is based on local network-state information for path selection. Thus, each router only distributes information about the load on each link to only its immediate neighbors.

With a similar approach, MRDV [3] does not require the exchange of any load information: each router running MRDV algorithm allows non-optimal paths to be used according to a variance factor reflecting the load on the next hop. Consequently, a MRDV router only has to monitor the load on its own links and can coexist with non-MRDV routers in the network. This approach is interesting for Ambient Networks environments due to both its decentralized scheme and the ability of its gradual introduction in networks allowing a smooth migration towards a full MRDV-enabled network. Next subsection briefly describes the MRDV basis.

2.1 Overview of Multipath Routing with Dynamic Variance (MRDV)

MRDV combines multipath routing with variance and distributed dynamic routing protocols. The core concept of the MRDV algorithm is that the number of alternative paths towards a destination depends on how occupied the links are. Multipath with variance routing algorithms allow traffic to each destination to be carried by other paths in addition to the paths with the minimum cost if the comparison between its metric and a threshold meets the following rule:

$$M \leq M_{min} \cdot V \tag{1}$$

where M is the metric of the path, M_{min} is the metric of the optimal path and V is the variance parameter of the output interface towards the next hop in the optimal path.

MRDV adjusts the variance parameter dynamically, according to the average load that the router detects in the next hop of the optimal path towards the destination. A different variance is defined for each output interface: every router monitors load in its adjacent links and modifies the variance of those interfaces according to their load.

According to the variance, new paths will be considered as suitable: load is distributed among these suitable paths, but the traffic offered to every path is inversely proportional to the path cost, so that the less cost a path has, the more traffic it receives. MRDV distributes traffic properly even when not all the interfaces are overloaded. In this case, only these overloaded links overflow traffic to other interfaces. Therefore, this algorithm is decentralized and IP compatible, and also adds the ability to adapt the variance to the traffic demand automatically.

With this approach, every router reacts to its own view of the network state: the average load of its adjacent links. The forwarding decisions are only based on local information and not on global information, as happens with other routing solutions that modify link costs according to the network status. However, two issues must be considered to prevent instability problems in MRDV. First, the variance must describe a hysteresis cycle, where relative increments in variance are proportional to relative increments in average load. Considering that the minimum variance is 1 (ECMP situation), the expression will be the following:

$$\left.\begin{array}{l} \frac{\partial V}{V} = K\frac{\partial \rho}{\rho} \\ V(\rho = 0) = 1 \\ V(\rho = 1) = V_{max} \end{array}\right\} \Rightarrow V = 1 + (V_{max} - 1) \cdot \rho^{K} \qquad (2)$$

where K is any real positive number and a design parameter, and V_{max} is the maximum possible variance.

Therefore, the hysteresis cycle is defined by the values of K for each of the two sections (from now on, K_{up} for the ascending curve V_{up}, and K_{dn} for the descending curve V_{dn}) and a common parameter V_{max} for the maximum variance. These parameters define the behavior of the algorithm. For simplicity, $K_{up} = 1/K_{dn}$ is proposed.

The other key issue regarding MRDV stability is the choice of the frequency to refresh the variance parameter as a trade-off between response time and accuracy in measures. Based on our experience with MRDV simulations, the update interval should never be less than about ten seconds, since a shorter update interval could lead to a too unstable behavior in the presence of bursty traffic.

MRDV has been implemented in Network Simulator 2 (ns-2) [9] and evaluated in different scenarios. Detailed results can be seen in [3], where MRDV is compared with OSPF without and with ECMP. In a realistic scenario with a typical backbone topology composed of 12 nodes and traffic with different burstiness degrees, the network is able to carry around 35% more traffic with MRDV than OSPF without ECMP, and around 15% more than OSPF with ECMP. In spite of these promising results, routing loops were affecting negatively to the traffic performance in these simulations. Thus, a mechanism to avoid them was considered as a key requirement for a satisfactory traffic management solution.

3 Description of Loop Avoidance Protocol (LAP)

In order to develop an algorithm for avoiding loops, a distinction between primary and secondary loops has been made. Primary loops, as Fig.1.a shows, appear when a node A tries to introduce a new sub-optimal path to reach D through B, which has A as the next hop of the optimal path to D.

Secondary loops are shown in Fig.1.b and Fig.1.c. In the first one (primary path sees a secondary one), there are both an optimal path (from B to A, however B has not A as its next hop in the optimal path to D) and also a secondary one (from A to B) to reach the same destination. In Fig.1.c (secondary path sees a secondary one), a loop is caused by two secondary paths, each one with its own percentage of routed traffic, α and β.

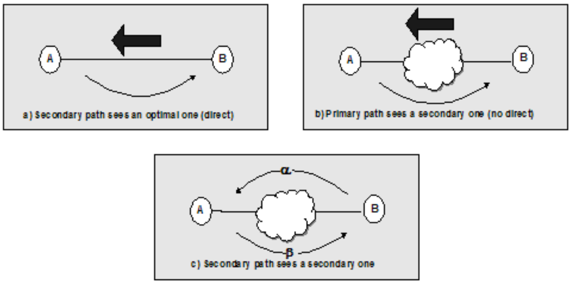
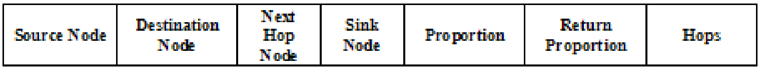

A
B
a) Secondary path sees an optimal one (direct)
A
B
b) Primary path sees a secondary one (no direct)
α
A
B
β
c) Secondary path sees a secondary one

Fig. 1. Types of loops



Source Node	Destination Node	Next Hop Node	Sink Node	Proportion	Return Proportion	Hops

Fig. 2. Structure of the LAPM. LAPM is composed of the next fields: *SourceNode* (id. of the node wanting to establish the secondary path), *DestinationNode* (id. of the destination node), *NextHopNode* (id. of the next hop of the secondary path *SourceNode* wants to establish to reach *Destination*), *SinkNode* (id. of the node starting the return phase), *Proportion* (direct proportion of the traffic sent by *SourceNode* to *Destination* through *NextHop* that reaches *SinkNode*), *ReturnProportion* (proportion of the traffic sent by *SinkNode* to *Destination* that reaches *SourceNode*) and *Hops* (number of hops that can be still leaped)

Taking into account this classification, two different mechanisms are proposed when a node is going to install a new secondary path: avoidance of primary loops and avoidance of secondary loops, described in Sections 3.1 and 3.2, respectively.

3.1 Avoidance of Primary Loops

Avoiding primary loops only requires a simple process to be computed at each router: when a router X is going to install a new sub-optimal path, if the candidate to new Next Hop (NH) to reach a destination has the router X as the next hop of its optimal path to reach the same destination, this new secondary path is discarded. Since X knows both the topology and the link-state information of the network, it is able to infer the optimal paths of NH by means of applying a Dijkstra algorithm [10] and no additional information exchange is required.

3.2 Avoidance of Secondary Loops

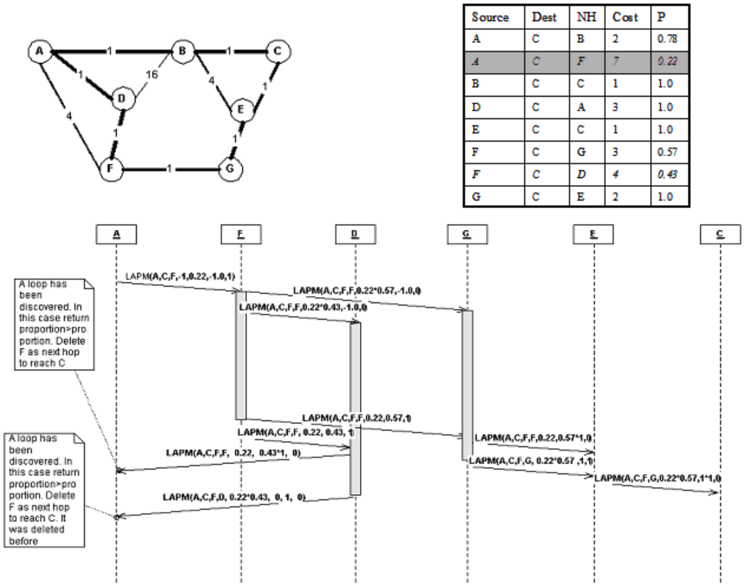
An information exchange is required in order to know whether a secondary loop will exist if the secondary path is installed and, if so, avoid it. We define the LAPM (Loop Avoidance Protocol Message) as the normalized message required for this information exchange, whose structure is shown in Fig.2.

This mechanism defines three main phases: a *forward phase* (calculation of the percentage of traffic routed by the forward path), a *return phase* (calculation of the percentage of traffic routed by the reverse path) and a *discovery* *phase* (triggered if a loop is discovered, where the secondary path is deleted if $Proportion$ is lower than $ReturnProportion$).

The *forward phase* is triggered by a node (N), trying to establish a new secondary path to a destination (D) by routing a traffic percentage (p) through the next hop (NH). This node sends a new LAPM to NH, initialized with the set of values (N, D, NH, -1, p, -1.0, $MaxHops$) according to the LAPM format defined in Fig.2. $MaxHops$ is a configurable parameter that defines the depth of the algorithm and represents a trade-off between loops avoidance and extra load in the network.

Once N sends the LAPM to NH, it is processed according to a defined set of actions to be triggered when a LAPM arrives.

When a node receives a LAPM, it firstly checks if the received message is a forwarding LAPM ($ReturnProportion$ is equal to -1). In this case, if $Hops$ is greater than zero, the node must resend the message to its next hops to reach


A
B
C
D
E
F
G
1
1
16
1
4
4
1
1
1
1
1
A
F
D
G
E
C
LAPM(A,C,F,-1,0.22,-1.0,1)
LAPM(A,C,F,F,0.22*0.57,-1.0,0)
LAPM(A,C,F,F,0.22*0.43,-1.0,0)
A loop has been discovered. In this case return proportion>proportion. Delete F as next hop to reach C
LAPM(A,C,F,F,0.22,0.57,1)
LAPM(A,C,F,F, 0.22, 0.43, 1)
LAPM(A,C,F,F,0.22,0.57*1,0)
LAPM(A,C,F,F, 0.22, 0.43*1, 0)
LAPM(A,C,F,G, 0.22*0.57 ,1,1)
LAPM(A,C,F,G,0.22*0.57,1*1,0)
A loop has been discovered. In this case return proportion>proportion. Delete F as next hop to reach C. It was deleted before
LAPM(A,C,F,D, 0.22*0.43, 0, 1, 0)

Source	Dest	NH	Cost	P
A	C	B	2	0.78
A	*C*	*F*	*7*	*0.22*
B	C	C	1	1.0
D	C	A	3	1.0
E	C	C	1	1.0
F	C	G	3	0.57
F	*C*	*D*	*4*	*0.43*
G	C	E	2	1.0

Fig. 3. Example of the avoidance of secondary loops. The figure shows a topology (top-left), the paths to reach C from all the nodes in the network (top-right) and a sequence diagram with all the messages exchanged by the nodes in the network when A wants to establish a new secondary path to reach C (bottom). In this example, we can distinguish the forward phase (A sends a new LAPM to F, F resends this message to its next hops with updated values of *Proportion* and so on), the return phase (e.g. when the timer expires, F sends to all its next hops to reach C a new LAPM with the initialized value of *ReturnProportion*) and final the discovery phase (A receives a LAPM with *Source* equals to A, and compares the values of the *Proportion* fields and deletes from its routing table F as a possible next hop to reach C).

D with updated values of *Proportion*, taking into account the percentage of traffic that the node routes through each one, p_i; therefore, for each next hop to reach D, the node has to resend the received LAPM with updated values for $Proportion$ ($Proportion * p_i$) and $Hops$ ($Hops - 1$).

In addition to the sending the updated LAPM to its next hops, the node starts the *return phase*. In order to aggregate the forwarding proportions belonging to the same routing tree (whose key is defined by $SourceNode$, *Destination* and $NextHop$), each node must maintain a list with the sum of the *Proportion* fields received in different LAPMs for the same routing tree. Consequently, when the return phase starts, if there is another registry in the list for that routing tree, the value of its proportion is updated (the received *Proportion* is added to the stored value). If not, a new registry is added to the list with the values included in the received LAPM, and a timer is triggered for that registry. When this timer expires, the node sends a new LAPM to each next hop of its routing table to reach D. The node initializes a new LAPM with the values stored in the list for $SourceNode$, *Destination*, $NextHop$ and *Proportion* and for *ReturnProportion*, $SinkNode$ and $Hops$ it uses p_i (traffic proportion routed to reach D from the specific next hop), the identifier of the node and the configurable parameter $MaxHops$, respectively.

On the other hand, when a returning LAPM is received (*ReturnProportion* different from -1), the node checks if the value of $SourceNode$ is equal to its own node identifier. If this condition is met, a secondary loop has been discovered and the discovery phase starts. Otherwise, if $Hops$ is greater than zero, the return phase continues and the LAPM is resent to all the next hops to reach D by updating the values of *ReturnProportion* ($ReturnProportion * p_i$, where p_i is the proportion of traffic sent by this hop) and $Hops$ ($Hops - 1$) fields.

Similarly to the return phase policy, in the *discovery phase* each node must also maintain a list to manage the received return LAPMs containing information about *Destination*, initial $NextHopNode$, $SinkNode$ (that one that initialized the return phase), *Proportion*, *ReturnProportion* and a timer to check if the path must be deleted. Therefore, when a loop is discovered, the node firstly checks if *ReturnProportion* of the received LAPM is greater than the *Proportion* contained in the same message. In this case, the node deletes from its routing table the $NextHop$ to reach *Destination*. If this is not the case, and there is another registry in the list with the same values of *Destination*, $NextHop$, *Proportion* and $SinkNode$, the value of *ReturnProportion* is updated by means of adding the just-received *ReturnProportion*. If not, the node introduces a new registry in the list with the values received in the LAPM and the initial value of the timer (also proportional to the $MaxHops$ configuration parameter). When the timer expires the node checks if fixed *Proportion* is equal or lower than the store, and maybe updated, *ReturnProportion*. In this case, the secondary path to reach *Destination* through $NextHop$ is deleted.

Fig.3 shows how the phases defined above converge and allow a router wanting to establish a new secondary path to avoid loops.

4 Evaluation of the Proposal by Simulation

The proposed solution for intra-domain traffic management in Ambient Networks, (MRDV+LAP), has been implemented in Network Simulator 2 (ns-2) [9] in order to evaluate by simulation the efficiency of the proposal. Two different topologies have been used in these simulations: a basic low-meshed topology with seven nodes and a more realistic and meshed topology with twelve nodes.

The traffic pattern used to feed these topologies is composed of both TCP and UDP traffic. For each pair of nodes in the network, each node contains at least one FTP application (and sends/receives TCP traffic to/from the other nodes in the network) and both constant and exponential bit rate applications are established. All the nodes send traffic rate to all the other nodes in the network, which is multiplied by a scale factor to increase the traffic load level, as it can be seen in further graphs.

In order to evaluate the performance perceived by this traffic, the evolution of the loss ratio and the mean delay for the UDP traffic and the average throughput obtained for the FTP applications have been analyzed. Moreover, the evolution of the loop probability has been monitored in order to analyze the efficiency of LAP.

For each simulated traffic level, ten simulations with different seeds have been performed, in order to estimate the error associated to a given statistical confidence interval. The further graphs show the mean value, while the highest values of the errors for a confidence interval of 90% are given in the figure captions for reference.

The following subsections present these scenarios and analyze the obtained results.

4.1 Basic Scenario

Firstly, we have evaluated the performance of the protocol in a basic scenario, whose topology is shown in Fig.4. The traffic matrix of this scenario is defined according to the principles explained above; however, in order to simulate a typical interconnection point to external networks, traffic rates of UDP flows sent or received by node 1 are increased a 30%. Regarding FTP applications, the file sizes sent by the sources depend on the traffic level of each mechanism.

This scenario has been simulated with different routing options: OSPF with ECMP (ECMP), ECMP with MRDV (MRDV), MRDV avoiding primary loops,

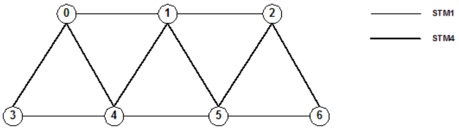


Fig. 4. Topology used in the basic scenario. Link delays set to 5 ms, link capacities to STM1 (155 Mbps) and STM4 (622 Mbps).

and MRDV with full LAP up to 1-hop loops. It has been monitored the UDP loss ratio, the TCP throughput and the loop probability and looped traffic obtained in each option. Fig.5 shows the mean values of these parameters.

Due to the usage of basic MRDV with respect to ECMP, it can be seen that UDP losses are postponed with MRDV and the congestion point for TCP traffic also appears later. For example, if we compare the traffic level when the loss ratio in both options overpasses a threshold of 1%, basic MRDV can manage around 38% more traffic than ECMP. Moreover, for a traffic level resulting in a loss ratio for UDP traffic of 1% in the case of MRDV, ECMP obtains 2.5% of losses. For this same traffic level, the average number of bytes received by a FTP application is 19.5% higher in the case of MRDV with respect to ECMP. As these results show, the use of MRDV can significantly improve the performance in the network for both UDP and TCP traffics.

The simulation results also show how LAP clearly improves the MRDV performance with respect to the basic MRDV option: for the same traffic level resulting in a loss ratio of 1% in MRDV, the avoidance of primary loops reduces to 0.19% the loss ratio and with the use of LAP with one hop, 0.18%. This small difference is due to the low possibility of routing loops with more hops since the topology is low-meshed. Looking at the obtained the loop probability in each case, it is reduced from 4.6% to 0.9%.

If we compare the traffic level when the loss ratio exceeds a 1%, LAP allows to duplicate the traffic level carried by the network. Finally, regarding the performance of TCP traffic, the congestion point, located where the number of received packets decreases for higher traffic levels, appears with a traffic increment of around 33% in the case of MRDV+LAP with respect to the congestion point of ECMP.

As expected, loop probability has been decreased and is even maintained when the traffic level is increased, as Fig.5 shows. This is because, although new routing loops appear, LAP discovers them and removes them from the routing tables. The remaining loops are those with more hops, which LAP is not considering in exchange of introducing less traffic overhead in the network.

4.2 Realistic Scenario

We have also evaluated the performance of MRDV with LAP in a more meshed topology (shown in Fig.6) based on the core network of the reference transport network presented in the IST-LION project [11].

In order to make TCP traffic more realistic, five Edge Nodes (EN) have been attached to a Core Node (CN) with links of 1ms delay and 10 Mbps capacity. These ENs run FTP applications and establish TCP connections with other ENs in such a way that fifteen FTP transactions running during the whole simulation time are established between each pair of CN nodes, that is, three per each EN associated. Regarding the UDP traffic, a rate proportional to the population of both source and sink cities has been set between the CNs.

Fig.7 presents the obtained simulation results, where MRDV without any mechanism for avoiding loops performs worse than ECMP due to the high ap-

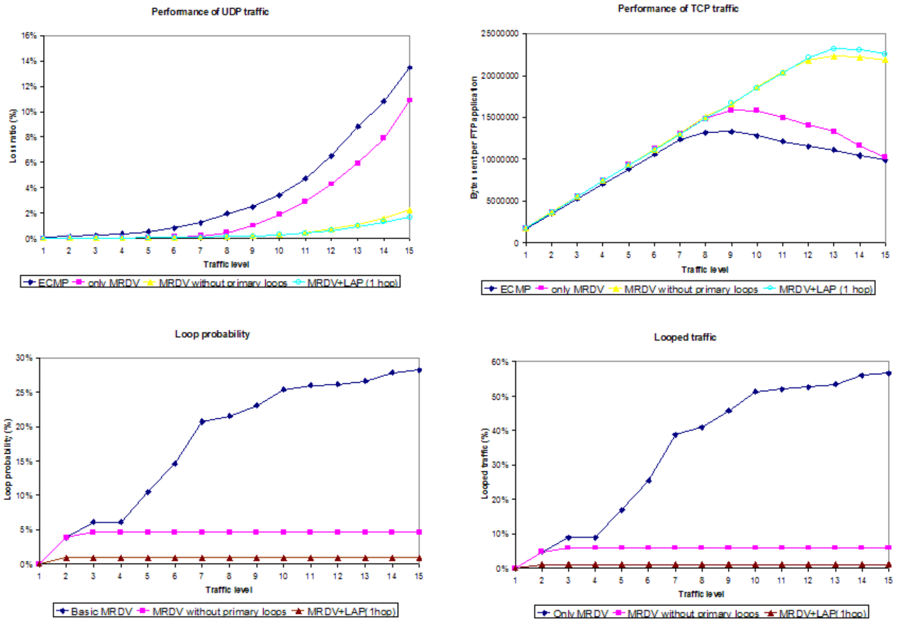


Fig. 5. Results obtained from the basic scenario. Top-left: Loss ratio of UDP traffic (max. error: 2.5%). Top-right: throughput of TCP traffic (max. error: 1.3%). Bottom-left: loop probability. Bottom-right: looped traffic.

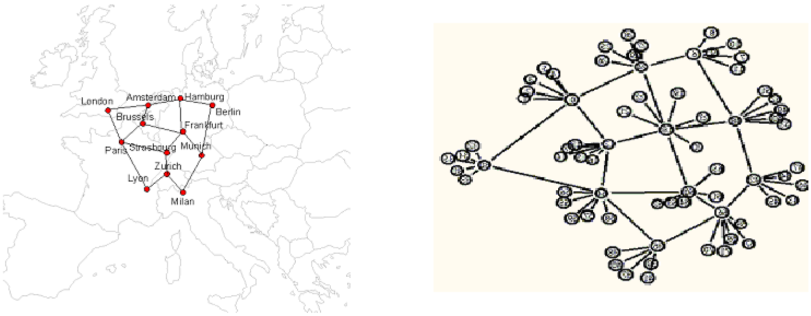


Fig. 6. Topology for the realistic scenario: backbone topology used (left) and topology of the simulated scenario, including TCP traffic sources and sinks (right)

pearance of loops, even with low traffic levels. As it can be seen, loop probability rapidly increases and goes over 50%. Therefore, this scenario justifies the use of mechanisms for avoiding loops.

Nevertheless, the results also show how the use of LAP can significantly improve the performance of MRDV. Specifically, the avoidance of only primary loops, for a traffic level value that causes a loss ratio of 5.8% for UDP traffic, MRDV+LAP obtains a loss ratio of 2.9%, reducing UDP losses around 50%. If

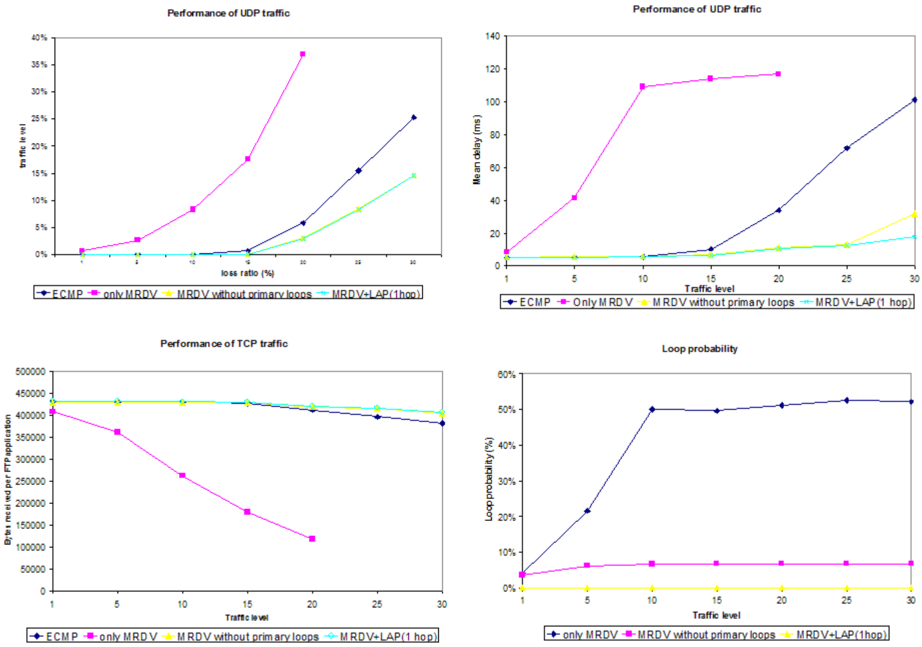


Fig. 7. Results obtained from the realistic scenario. Top-left: Loss ratio of UDP traffic (max. error: 0.4%). Top-right: Mean delay of UDP traffic (max. error: 0.6% and 9% for MRDV and ECMP respectively). Bottom-left: Throughput of TCP traffic (max. error: 0.2%). Bottom-right: loop probability.

we compare the traffic level when the loss ratio in both options overpass a 5%, MRDV with LAP can manage around 15% more traffic than ECMP. However, if we evaluate the TCP performance for the same traffic level (20), the throughput only increases by 2% due to the use of MRDV+LAP instead of ECMP. It is needed a higher traffic level to obtain significant benefits; e.g. for a traffic level of 30, this benefit is around 8%.

The results obtained with one-hop LAP are not significantly better than those ones with just avoiding primary loops because most of the loops are primary ones. In fact, the avoidance of primary loops reduces loop probability to 6%. Anyway, one-hop LAP removes all the existing loops.

5 Conclusions and Further Steps

This paper presents the combination of MRDV and LAP as a satisfactory solution for the traffic management in Ambient Networks. The proposal presents a decentralized friendly-migrable solution to distribute traffic load in the network thanks to the use of MRDV, whose performance is improved by LAP by successfully removing secondary paths that can cause loop appearance. Specifically, the use of LAP is interesting in those scenarios where loop probability is very high, as it has been shown in the realistic scenario.

Moreover, at least in the studied scenarios, only with the avoidance of primary loops, traffic performance is significantly improved. So, the second phase of LAP would not be necessary since the improvement in the traffic performance is negligible. Nevertheless, more simulations with different scenarios are needed to have more confidence in this conclusion.

Another issue that must be analyzed in further steps is the behavior of MRDV+LAP in case of variations during the simulation time in both the traffic matrix (sudden changes in traffic patterns) and the network topology (link and node failures). Also, we have to evaluate how configuration of timers used in the return and discovery phases of LAP can affect the performance of the protocol.

References

[1] Niebert, N., Schieder, A., Abramovicz, H., Malmgren, G., Sachs, J., Horn, U., Prehofer, C., Karl, H.: Ambient Networks: An Architecture for Communication Networks Beyond 3G, IEEE Wireless Communications, Abril 2004.

[2] Awduche, D., Malcolm, J., Agogbua, J., O'Dell, M., McManus, J.: Requirements for Traffic Engineering Over MPLS, Request for Comments 2702, Internet Engineering Task Force (1999).

[3] Ramón-Salguero, F.J., Enríquez-Gabeiras, J., Andrés-Colás, J., Molíns-Jiménez, A.: Multipath Routing with Dynamic Variance, COST 279 Technical Report TD02043 (2002).

[4] Hopps, C.: Analysis of an equal-cost multi-path algorithm, Request for Comments 2992, Internet Engineering Task Force (2000).

[5] Moy, J.: OSPF Version 2, STD 54, Request for Comments 2328, Internet Engineering Task Force (1998).

[6] Callon, R.: Use of OSI IS-IS for Routing in TCP/IP and Dual Environments, Request for Comments 1195, Internet Engineering Task Force (1990).

[7] Villamizar, C.: OSPF optimized multipath (OSPF-OMP), Internet Draft, draft-ietf-ospf-omp-03 (1999).

[8] Gojmerac, I., Ziegler, T., Ricciato, F., Reichl, P.: Adaptive Multipath Routing for Dynamic Traffic Engineering, IEEE Globecom (2003).

[9] The Network Simulator - ns-2, http://ww.isi.edu/nsnam/ns.

[10] Cormen, T.H., Leiserson C.E., Rivest R.L.: Introduction to Algorithms, MIT Press (1990).

[11] Reference Transport Network Scenarios, IST COST/COST 266, http://www.ibcn. intec.rug.ac.be/projects/IST/NRS.

Performability Analysis of an Adaptive-Rate Video-Streaming Service in End-to-End QoS Scenarios*

I.V. Martín, J.J. Alins, Mónica Aguilar-Igartua, and Jorge Mata-Díaz

Telematics Engineering Department, Technical University of Catalonia (UPC), Jordi Girona 1-3,
08034, Campus Nord, Barcelona, Spain
{isabelm, juanjo, maguilar, jmata}@entel.upc.es.

Abstract. Nowadays, dynamic service management frameworks are proposed to ensure end-to-end QoS. To achieve this goal, it is necessary to manage Service Level Agreements (SLAs), which specify quality parameters of the services operation such as availability and performance. This work is focused on the evaluation of Video-on-Demand (VoD) services in end-to-end QoS scenarios. Based on a straightforward Markov Chain, Markov-Reward Chain (MRC) models are developed in order to obtain various QoS measures of an adaptive VoD service. The MRC model has a clear understanding with the design and operation of the VoD system. In this way, new design options can be proposed and be easily evaluated. To compute performability measures of the MRC model, the randomization method is employed. Predicted model results fit well to the ones taken from a real video-streaming testbed.

1 Introduction

During the last years, Video-on-Demand (VoD) applications for the transmission and distribution of video have experienced a growing development and acceptance from the users. Video-streaming systems have a special relevance in wired and wireless networks. In these systems, the video is distributed for its reproduction in real-time [1]. The video server of a video-streaming system stores a set of movies that can be requested by any client. If the connection request is accepted, a session is initiated; then a multimedia stream flows through a set of heterogeneous networks from the video server to the client terminal.

In end-to-end Quality of Service (QoS) scenarios, QoS measures such as packet loss, packet delay and jitter must be guaranteed when the connection is accepted. These real-time guarantees required by the VoD systems could be achieved using QoS differentiation between traffic classes over heterogeneous networks. On the other hand, with the aim of reducing the huge amount of information generated by the video source, loss compression techniques are applied. The most common coding techniques are H.26x and MPEG standards [2]. The price to pay for a high compression level is a degradation level in the image quality.

VoD systems with QoS guarantees might be designed to provide a uniform level of image quality to their users. The video flows coded with a constant image-quality

* This work has been financed by the Spanish investigation projects DISQET (CICYT TIC2002-00818), CREDO (CICYT TIC2002-00249) and ARPA (CICYT TIC2003-08184).

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 157–168, 2005.

present a high variability in their transmission bit rate. In this way, the amount of network resources required for the transmission fluctuates notably.

The adaptive VoD services employ a set of policies for dynamic resource allocation. It is accomplished by means of signalling protocols used between the service and the network. Thus, related to the bit rate variability of the flow, the service raises renegotiations to the network in order to modify the allocated resources during the session. These renegotiations are performed at the temporal-scale of the scenes in a video sequence. In this way, the amount of network resources reserved during the session are reduced substantially, and a more efficient exploitation of these resources is achieved [1,3,4]. Therefore, the number of concurrent streaming sessions in the system is incremented. However, the image quality will be reduced in some congestion moments when the flow with the selected quality cannot be transmitted. In these congestion situations, the service adapts the transmission bit rate to the available network resources applying a higher compression level or managing the enhanced layers when scalability techniques are employed [1]. Thus, the final QoS provided to the customers of these streaming services depends on the available network resources.

Both the service providers and the customers are indeed interested in tools which support to quantify the performance of these systems from their points of view. Analytical tools are the most appropriate mechanisms to facilitate the required evaluation. Moreover, these tools should provide feasibility to incorporate modifications into the system in an easy way. Further, they also must admit a computational evaluation. This kind of analytic tools help to address some of the typically required main objectives: to maximize the use of network resources and the QoS offered to the users and, to define billing metrics. Likewise, these tools could compute diverse parameters in order to specify, to manage and to control the fulfilment of the Service Level Agreements (SLAs). The management of SLAs is a current challenge into the multimedia services area. Further, it has a great commercial interest. There are diverse recent proposals about SLA management (e.g. [5,6]), although none of them specifies how to quantitatively evaluate the user-level SLA parameters.

One of the main objectives addressed in the present work is to compute *a priori* the QoS offered to the user of a video-streaming application. In particular, we are interested in the evaluation of adaptive VoD systems, in which video sources are capable to adapt their output bit rate to the time-varying available network resources.

Some proposals of design and evaluation of adaptive VoD systems are presented in [3,7,8,9,10]. However, most of these proposals use either simulation models or real platforms to carry out the performance evaluation of these systems. These evaluation techniques hinder the system analysis and also the study of several design options. In addition, some analytical proposals do not regard the interaction between the different video sources sharing the network resources. On the other hand, works focused on characterizing and modelling a single video flow [4,11,12] are not enough to evaluate an adaptive VoD session because they do not take into consideration the dynamism of the video quality changes all over the transmission of the stream.

In [13] we proposed a generic method to develop Performability models [14] for VoD systems. This method solves the lacks above mentioned. The applicability of this method is based on the characterization of the coded multimedia flows and the channel behaviour. This characterization requires suitable markovian models of both the flows and the channel. In addition, an analytical model developed with this generic method for a VoD service was presented in [13]. This model provides accurate results for the measures of user-level QoS parameters such as the image quality, reserved resources, or effectively-used resources. However, a problem of this model is that the computational cost may increase dramatically when the amount of accepted connections or the amount of user-classes increase.

In the present work, we obtain other two new analytical models based on the method presented in [13]. These models reduce the states space to characterize the resources reserved by a group of users. Using these proposed models, we have analyzed the performance of a VoD service varying some design parameters.

The rest of the work is organized as follows. Section 2 describes the evaluated VoD system. In section 3, a background of the previous work presented in [13] is summarized. In section 4 we propose two new simpler analytical models of the adaptive VoD service. Next, in section 5 some numerical results evaluating both VoD models are shown. The results of these models are compared with experimental measurements obtained from the SSADE project (*http://ssade.upc.es*) implemented by the Telematics Services Research Group of the Polytechnic University of Catalonia. Finally, conclusions and future work are presented in section 6.

2 The System Description

Figure 1 depicts the VoD system analysed in this work. Video sequences have previously been coded using the VBR MPEG-II algorithm and then, stored in the video server. When any customer of the VoD service demands one of these sequences, a connection is established if the video service has enough resources to provide the contracted user's profile, i.e. the agreements specified in the service contract. In many IP QoS-aware networks, the RSVP (Resource reSerVation Protocol) is employed as signalling protocol to manage resource reservation requests [15]. Video-streaming services that use RSVP send requests to the network in order to adjust the required resources of the video-stream transmitted. The description of these resources is specified by means of the *Traffic Specification* (TSpec) parameters carried in the PATH messages of the RSVP. The VoD server requests resource reservation for each session dynamically, i.e. the amount of required resources are calculated and adapted for smaller intervals than the length of the sequence. So, different network resources are requested over the whole transmission of a sequence. Moreover, this renegotiation process yields that the available network resources change due to the interaction between the multiplexed connections. With the aim that the video sources are capable to adapt their output bit rate to the time-varying network resources, each available sequence has a set of MPEG flows coded with different quantization step (Q). Then, each available flow offers a different image quality according to Q [2]. For each accepted session, the transmitted stream will match with one of the different available coded flows of the requested sequence. This selection changes depending on the image quality contracted by the user and on the result of the reservation request produced by the end-to-end admission control of the RSVP-based system.

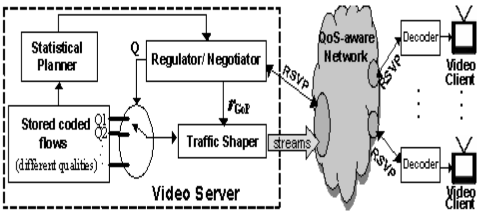


Fig. 1. System model for the VoD service

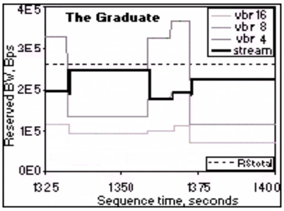


Fig. 2. Required bit rate

To carry out the system functions, three blocks have been designed as it is sketched in Fig. 1. These blocks are performed as follows. For each available flow, the *Statistical Planner* block has previously calculated and stored the TSpec parameters of each scene and the events of resources renegotiation in each sequence. When scene changes or variations of the available resources happen, the *Regulator/Negotiator* block decides which flow (Q_i) will be transmitted. To guarantee a minimum video quality, the minimum reservation needed to transmit the lower image quality flow must always be assured. The *Traffic Shaper* block extracts the variability introduced by the frame coding modes (Intra, Predicted and Bidirectional-Predicted) of the MPEG algorithm. In this way, the bit rate is smoothed and it is maintained constant (r_{GoP}) for a GoP (Group of Pictures) interval.

As an example of the transmitted stream, Fig. 2 shows the bit rate required to transmit the sequence "The Graduate" coded with a quantification step Q equals to 4, 8 and 16. The dark line shows the bit rate reserved to the connection, when the total resources for the video service are 270,000 Bytes per second. Notice that this stream matches with one of the available flows for each moment of transmission time.

3 Background

3.1 Scene-Based Markovian Models for a Video Sequence

In order to efficiently characterize the network resources required by a constant-quality flow of a video sequence, we need to identify the groups of frames with same complexity or activity in the sequence. The identification process of these consecutive groups of frames has been called in the literature as *Segmentation* [16]. The segmentation of a video sequence results on series of groups of pictures with similar requirements of network resources [17]. These segments, also named scenes, define different complexity levels within the sequence. Through the classification of the scenes into activity levels, scene-based models have been proposed in previous works [18]. Some of the more relevant works have developed analytical models based on Markov chains. These models set the number of scene classes heuristically. Straightforward scene-based Markovian models represent scene changes by means of transitions between states, where states identify classes of scenes. For the sake of the simplicity, we will refer to each class of scenes as an activity level. An example of the Markovian scene-based model is shown in Fig. 3, where *L* activity levels are defined.

The segmentation process of different constant-quality flows of the same movie gives rise to the same scene bounds. Consequently, for a set of video-flowmodels

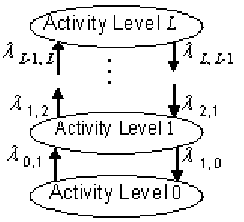


Fig. 3. Scene-based Markovian model

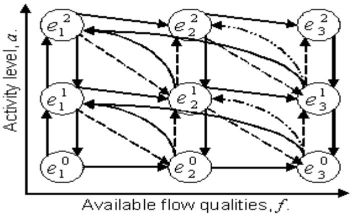


Fig. 4. Generic model of an accepted connection

related to the same movie, the changes of scene occur at the same time. Fig. 2 remarks how these flows are time-tracked. Hereafter, we will indistinctly refer to a change of the activity level of the sequence as well as a change of the activity level in anyone of its coded flows. In this work, the amount of resources required to transmit the scenes of each available flow are considered to be known and they have previously been calculated for all the stored sequences at the video server.

3.2 Generic Method to Develop Analytical Models of VoD Systems

A generic method to construct Markov Reward Chain (MRC) models for VoD systems was proposed in [13]. Finally, to carry out computations of QoS measures, the method of Randomization [19] was applied to the MRC. The generic modelling methodology presented there consists on 5 steps that obtain a MRC that statistically characterizes the network resources required by a connection and the available resources for this connection. Also, applying this methodology, an analytical model of a particular VoD system was derived in [13]. For more information of this model and the generic method, please refer to [13]. Below, we summarize this model and later on in section 4 we propose some modifications to this model obtaining new models which reduce the space of states and provide a faster computation of expected results.

In **Step 1**, a markovian model for each available fixed-quality video flow in the VoD system is derived in a similar way as it was developed in [3]. This model is composed by three states that define three activity levels (as it is presented in section 3.1).

The Continuous-Time Markov Chain (CTMC) shown in Fig. 4 is an example derived from **Step 2**. For the sake of the clarity, only 3 different constant image-quality flows have been depicted in the draw. Each column corresponds with the model of each available flow, which was obtained in step 1. This CTMC models the behaviour of a connection in the system, where e_f^a is a connection state where flow of quality f (1: worst quality... F: best quality) in the activity level a (0: regular, 1: medium, 2: high) is transmitted through the connection. The transitions between states reflect the scene changes and renegotiation decisions that have been designed in the VoD system. In addition, in this example while the stream remains in the same activity level, the system periodically tries to improve sending requests for the next better image-quality. We call this process *polling of improvement*. The transition rates in this CTMC depend on the required resources and on the available resources. In order to formally express these dependencies, two boolean factors are defined:

$$\alpha(e_f^a) = \begin{cases} 1 \text{ , } if\, RSV(e_f^a) \leq RS_{available} \\ 0 \text{ , } otherwise. \end{cases} \tag{1}$$

and

$$\beta(e_f^a, e_g^b) = \begin{cases} 1 \text{ , } if\, RSV(e_f^a) \leq RSV(e_g^b) \\ 0 \text{ , } otherwise. \end{cases} \tag{2}$$

where $RS_{available}$ is the amount of available resources for the connection and $RSV(e_f^a)$ is the amount of network resources that the system reserves for the connection when the server transmits a flow of quality *f* in activity level *a*.

Let $\Psi(e_f^a, e_g^b)$ be the transition rate from state e_f^a to state e_g^b for a connection. These rates in the CTMC, related to increments or decrements of the activity level in the video sequence, are formulated by means of equations which depend on the rates $\lambda_{a,b}$ and the factors β (see [13]). Finally, transitions owing to the *polling of improvement* are formulated as a function that depends on the poll rate λ_p and the factor α. Note that the factors α and β provide the general characterization of the adaptive VoD service by means of the proposed CTMC. Different behaviours for the renegotiation mechanism can simply be designed using these factors. When several connections are being served, $RS_{avalaible}$ vary during the time. In this case, it is mandatory to know the state of all the connections that interact with the one under analysis.

In **Step 3** the model of *N* accepted sessions is achieved. Firstly, the state of all the connections with a same QoS profile is defined. In this case, that we called homogeneous case, each connection is characterized with the same parameter values of the generic connection model that has been described in step 2. Let $S^* = \{(n_1^0, n_1^1, n_1^2), (n_2^0, n_2^1, n_2^2), \ldots, (n_F^0, n_F^1, n_F^2)\}$ be the system state, where each component n_f^a is the number of connections transmitting a flow of quality *f* in activity level *a*. Let $S_{+(g,b)}^{-(f,a)}$ be a system state with one more connection transmitting a flow of quality *g* in activity level *b* and one less connection transmitting a flow of quality *f* in activity level *a*, regarding state S^*. If all transmissions are uncorrelated, only transitions from state S^* to state $S_{+(g,b)}^{-(f,a)}$ can occur. Then, the transition rates $\Psi(S^*, S_{+(g,b)}^{-(f,a)})$ are expressed as a function of rate ψ multiplied by n_f^a. Where, generalizing (1) for *N* connections and taking into account a conservative admission control, the factor $\alpha(S^*)$ is expressed as follows:

$$\alpha(S^*) = \begin{cases} 1 \text{ , } if\, \sum_{\forall f} \sum_{\forall a} n_f^a \cdot RSV(e_f^a) \leq RS_{total} \\ 0 \text{ , } otherwise. \end{cases} \tag{3}$$

To develop the case of heterogeneous customers, the simpler way is to define the system state as the joining state of each user class, this is: *S* = $S^1, S^2, \ldots, S^C$.

Step 4 is applied to evaluate the performance of a session in the system with other *N* accepted sessions. Let e_f^a, S be a system state, where e_f^a describes the state of the connection under evaluation and S characterizes $RS_{available}$. The state *S* may be described e.g. as the one obtained in step 3 or any markovian model of channel capacity.

4 The New Analytical Models

The analytical model we presented in [13] provides accuracy results for the performance measures of the VoD system. Moreover, with this model we accomplish the performance computations and system modifications in an easy way. Nevertheless, as N grows, this model increases combinatorially the number of states and therefore the computational cost increases in the same way [20]. Several methods have been proposed to improve the evaluation efficiency for models with these characteristics [19]. However, these methods may be insufficient to reach the objective of developing a tool to compare different design options of the system at a low calculation time (real time, if possible). To address this issue, we have investigated how to reduce the states space of the model to evaluate the QoS provided to a customer of the VoD system.

In our VoD service, a characterization of the resources reserved by the other N sessions accepted in the system is necessary to provide the available resources for the session under evaluation, $RS_{available}$. This characterization can be reached with the analytical model of N accepted sessions depicted in step 3 of section 3.2, where the states space equals to $(N+3F-1)!/(N!(3F-1)!)$ for F available video-qualities and each ones modelled with 3 activity levels. Further, after applying the step 4 to evaluate the session under analysis, the states space of the model is equal to $3F \cdot (N+3F-1)!/(N!(3F-1)!)$. We have reduced to two states the model of each available flow (see Fig. 3) for the N sessions that are not under analysis. Applying this reduction of states, the states space in step 4 is reduced to $3F \cdot (N+2F-1)!/(N!(2F-1)!)$ for the two models proposed following (see Table 2).

4.1 Analytical Model 1

We have observed that, for any video sequence, the probability to achieve the highest activity level is very low for the model with three activity levels. Based on this observation, we propose to maintain the same regular activity level (level 0) and to establish just one state to define jointly the medium (level 1) and high (level 2) activity levels to model the resources required by each available flow of the sessions that are not under analysis. We do not need to carry out any new statistical analysis of the sequences to develop this new model. We only need to adjust the parameters of the new level 1. We called to this new level as level 1^*. The transition rates between level 0 and level 1^* are the same ones that between level 0 and level 1 in the analytical model with three levels. The resources required to be in level 1* are calculated as

$$RSV(e_f^{1^*}) = \frac{(RSV(e_f^1) \cdot time\_medium + RSV(e_f^2) \cdot time\_high)}{(time\_medium + time\_high)} \tag{4}$$

where $time\_medium = 1/(\lambda_{1,0} + \lambda_{1,2})$ and $time\_high = 1/\lambda_{2,1}$.

To characterize the resources reserved by the sessions that are not under analysis, we use the generic method where the model of each available flow includes the modifications above described. This way, we can characterize $RS_{available}$ included in the model of the session under analysis. This latter model is based on three activity levels to characterize each available flow.

We also have evaluated different options beside (4) to approximate $RSV(e_f^{1^*})$. For example, $RSV(e_f^{1^*})$ equals to $(RSV(e_f^1) + RSV(e_f^2))/2$ and equals to $RSV(e_f^1)$. But (4) has been the option that gives the most accuracy results.

We have compared numerical results of this analytical model with the ones of the original model presented in [13], showing a considerable improvement in the run-time (see Table 2) and the same accuracy. From the analysis of these numerical results, we have been able to adjust some parameters associated to each state leading to the following analytical model.

4.2 Analytical Model 2

We have observed that the regular level is the level with the longest scene length. Therefore, this level is the most influential in the evaluation measures. In addition, the standard deviation of the required resources of the regular level is higher than for the other levels. Then, we propose to modify the parameter $RSV(e_f^0)$ of the model as follows. For each flow in the regular activity level (level 0), a threshold is established to reach a new classification of the scenes in this activity level. Depending on the activity level of these scenes, we have the set of scenes which required resources are lower than the threshold and the set of scenes which required resources are higher than the threshold. We called these new sets of scenes as the lower and the higher sublevel of the regular level, respectively. The amount of required resources associated to the regular level may be any statistical measure of the required resources of all the scenes classified into one of the both sublevels: the lower or the higher. The selection of suitable statistical measures will lead to an analytical model that offers an lower bound or a upper bound for the evaluation measures of our VoD system. We call level 0* to this new regular level.

From the statistical study of different sequences, we have empirically established that best computation for the threshold is to average the required resources for all the scenes classified into level 0, weighted by their scene lengths. In addition, we computed this statistical measure in each resulting sublevel. With this choice, obtained results, compared to the ones from the real system, are very adjusted. However, we are taking additional mathematical methodologies into consideration for future works.

5 Numerical Results

In this section, numerical examples from the evaluation of the VoD system are presented. Several measures of the QoS offered to a customer of the VoD service have been computed, such as the PSNR, the failure time, the transmitted bit-rate and the reserved BW. The expected mean value of these measures for an observed user and the standard deviation between all the users are presented as well. The computation of these measures from the analytical models has been reached using the solution of the moments of cumulative reward presented in [20]. These results have been compared with experimental values from the SSADE video distribution testbed that provides an adaptive video-on-demand service. The configuration parameters of the VoD system used here are the same ones that in [13]. Likewise, the transmitted sequence is “The Graduate”, where the available flows are coded with quantization step Q equals to 4, 8 and 16. For simplicity, we refer to the model presented in [13] as **model_old** which parameters were defined in [13]. We use **model1** and **model2** to refer the two new analytical models presented in sections 4.1 and 4.2, respectively. Table 1 summarizes the values of the modified parameters used for **model1** and **model2**, this is level 0* and level 1* for each available flow of the sequence “The Graduate”. The lower sublevel of the regular level has been chosen as level 0*.

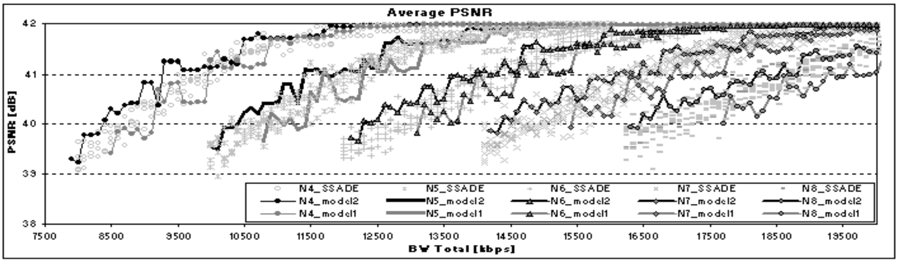


Fig. 5. Mean PSNR provided to a customer for the transmission of *"The Graduate"* movie

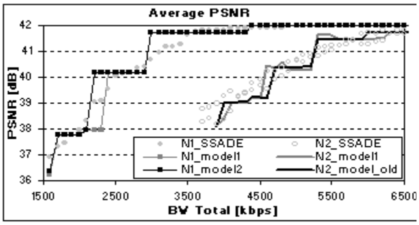


Fig. 6. Mean PSNR provided to a customer

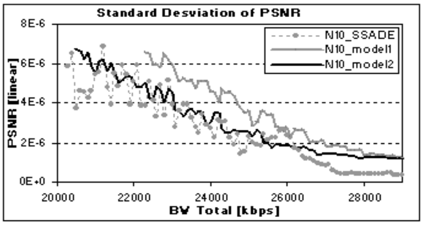


Fig. 7. PSNR standard deviation

Figures 5 to 7 illustrate some evaluation results of the proposed analytical models. The measurements taken from our testbed are pointed out as **SSADE**. Figure 5 depicts the *Mean PSNR* provided to a user and the standard deviation of this measure for all the accepted sessions. These figures are represented as a function of the total BW assigned to the VoD service and for a variable number of accepted streaming sessions (**N**). For each **N**, the curves start at the minimum BW required to accept the **N**th session in the VoD system. Discontinuities in the analytic curves are produced as a result of the discretization of the activity levels of the flows. Therefore, the performability results are discretized and softened as **N** grows. This happens since the connections are multiplexed and the values associated to the MRC states are softened as **N** increases. The light grey values (**SSADE**) depict measurements of each one of all the sessions accepted in our testbed. Note that, analytical curves of both the **model1** and the **model2** provide a good approximation as a lower and a higher bound with respect to the values from **SSADE**. Both the reserved and the transmitted rates present a similar behaviour (for space reasons they are not shown here).

The numerical results from the models are also sufficiently accurate to analyze the behaviour of the VoD service from the point of view of standard deviation. Fig. 7 presents the standard deviation for PSNR measurement of the 10 customers in the VoD service. Generally, the evaluation results from **model1** are very close to the ones from **model_old** for any value of **N** (e.g. see Fig. 6). However, run-time of 1st and 2nd moments for **model1** is considerable lower than run-time for **model_old,** as shown in Table 2, without losing precision. Another QoS measure that interests to both customers

Table 1. Estimated required resources[1] [bits/GoP]

	level 0*	level 1*
Vbr4	504715.88	782343.95
Vbr8	364476.06	421790.50
Vbr16	364476.06	364476.06

Table 2. Comparison of run-times[2] [seconds] and states space for **N** sessions, *F*=3, in the system

	model_old		model1 and model2	
N	#states	run-time	# states	run-time
5	4455	~30	1134	~5
8	218790	~950	11583	~60
12	1133730	~35000	55692	~740

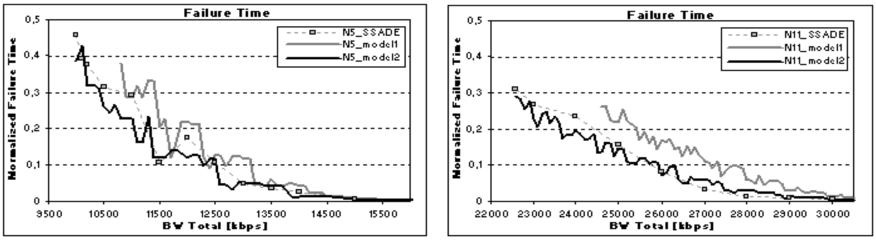


Fig. 8. Failure Total Time for 5 and 11 accepted sessions

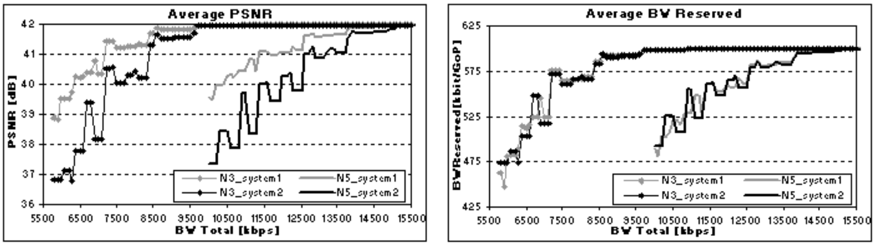


Fig. 9. Comparison between different VoD services (see text for explanation)

and service providers, is the total time when the service does not provides the committed QoS level to the customer. This measure can be interpreted as *failure time* (e.g. the service is in failure when the delivered flow is coded with a quality lesser than the video-quality contracted by the customer). Fig. 8 shows numerical results of this measure for a session for 5 and 11 accepted sessions, where a failure is defined when the service does not deliver the flow coded with Q=4 (i.e., the maximum video-quality available by the users of our VoD service) to the session under evaluation. In Fig. 8 the total failure time is normalized to the total time of the transmitted sequence. The **SSADE** curve depicts this measure for the last session accepted in the system. For a given total BW assigned to the video service, evidently the *failure time* increases as **N** increases. On the other hand,

[1] The level 1* is used by **model1** and **model2** and the level 0* is only used by **model2**.

[2] The solutions are implemented on Delphi 4 and executed on Pentium 1.4GHz, 768MB.

when the total BW assigned to the video service is the minimum such as **N** sessions can be accepted, we observe that the*failure time* decreases if **N** increases. For example; when **N**=5, total BW=10Mbps, the service is in failure approximately 40% of the total transmission time (see left graph in Fig.8). When **N**=11, total BW= 21Mbps, this time is 30% (see right graph in Fig. 8). Notice that the QoS provided by this VoD system never experiments a high degradation level. The designed user profile is such that each accepted connection can access to all the available resources, without distinction among the users. So the connections quickly acquire a high level of resources sufficient to reach the maximum quality. Therefore, the admission control designed for this VoD service provides suitable QoS guarantees and a strictly access to the system. In this way, once the connection is accepted, the user will perceive a video quality close to the maximum.

Other VoD system configurations or heterogeneous user profile yield other service behaviours. For example, Fig. 9 presents the *Mean* PSNR and reserved BW provided to a session in the system described above (pointed out as **system1** in Fig. 9) where three coded flows are available (Q=4, 8 and 16) and an other system which difference is that the flow coded with Q=8 is not available (pointed out as **system2** in Fig. 9). The curves in Fig. 9 verify that *mean* PSNR is lower in **system2** than in **system1** for the same reserved BW in both systems. We can establish that the VoD **system1** has a higher efficiency than the VoD **system2**, since it provides a more adaptive service.

6 Conclusions and Future Work

The end-to-end QoS provisioning in IP-based networks is a current challenge. Many QoS management frameworks available in the literature need additional mechanisms and procedures to quantitatively evaluate the user-level SLA parameters. The generic methodology proposed in [13] utilizes the characterization of the available constant-quality flows with the aim to develop an adaptive model for the video-streaming service. Models obtained with this methodology capture the dynamism of the transmitted stream taking into account the activity variations of the sequences and the interaction with the network. In the present work we propose two new analytical models of an adaptive VoD system employing this methodology. These models are simpler than the example presented in [13]. The computational cost is considerably reduced for both new models. The key of the goodness of these models has been to reduce the states space to characterize the resources reserved by a group of users. Different measures of performance evaluation of the VoD service have been calculated using these models. The precision of computed results allow us to verify that the developed models give a good estimation of the performance of adaptive multimedia systems in end-to-end QoS scenarios. The adjustment of the obtained results will depend on the suitable characterization of the multimedia sources and the transmission channels using suitable Markovian models. As future lines of research, we are studying others efficient ways to develop the heterogeneous problem, where diverse QoS customer profiles are considered. Likewise, well-known models in the literature can be integrated to characterize time-varying available resources produced by variations in wireless channel capacity.

References

1. Wu, D., Hou, Y., Zhu, W., Zhang, Y., Peha, J.: Streaming Video over Internet: Approaches and Directions. IEEE Trans. On Circuits and Systems for Video Technology **11** (2001)
2. Ghanbari, M.: Video Coding: An Introduction to Standard Codecs (IEE Telecommunications Series 42). IEE Publishing (1999)
3. De la Cruz, L.J., Mata, J.: Performance of Dynamic Resource Allocation with QoS Guarantees for MPEG VBR Video Traffic Transmission over ATM Networks. In: Proceedings of the IEEE GLOBECOM'99, IEEE Communications Society. (1999)
4. Manzoni, P., Cremonesi, P., Serazzi, G.: Workload Models of VBR Traffic and Their Use in Resource Allocation Policies. IEEE/ACM Transactions on Networking **7** (1999)
5. Cortese, G., Cremonese, P., D'Antonio, S., Diaconescu, A., Esposito, M., Fiutem, R., Romano, S.P.: CADENUS: Creation and Deployment of End-User Services in Premium IP Networks. IEEE Communications Magazine (2003)
6. IST Project: TAPAS– Trusted and QoS-Aware Provision of Application Services (2001) http://www.newcastle.research.ec.org/tapas/.
7. Muntean, G., Murphy, L.: A New Adaptive Multimedia Streaming System for All-IP Multi-Service Networks. IEEE Transactions on Broadcasting **50** (2004)
8. Lombardo, A., Schembra, G.: Performance Evaluation of an Adaptive-Rate MPEG Encoder Matching IntServ Traffic Constraints. IEEE/ACM Transactions on Networking **11** (2003)
9. Luna, C., Kondi, L., Katsaggelos, A.: Maximizing User Utility in Video Streaming Applications. IEEE Trans. on Circuits and Systems for Video Technology **13** (2003)
10. Ramanujan, R.S., Newhouse, J., Kaddoura, M., Ahamad, A., Chartier, E., Thurber, K.: Adaptive streaming of MPEG video over IP networks. In: Proceedings 22nd Annual Conference on Local Computer Networks, IEEE. (1997)
11. Adas, A.: Traffic Models in Broadband Networks. IEEE Communications Magazine **35** (1997)
12. De la Cruz, L.J., Fernández, M., Alins, J., Mata, J.: Bidimensional Fluid Model for VBR MPEG Video Traffic. In: 4th International Conference on Broadband Communications, IFIP, TC6/WG6.2. (1998)
13. Martín, I.V., Alins, J., Aguilar-Igartua, M., Mata, J.: Modelling an Adaptive-Rate Video-Streaming Service Using Markov-Rewards Models. In: Proc. of the First International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE04), IEEE. (2004)
14. Meyer, J. Teletraffic Science for Cost-Effective Systems, Network and Services, ITC-12. In: Performability Evaluation of Telecommunication Network. Elsevier Science Publishers B. V. (North Holland) (1989)
15. Bernet, Y.: RFC 2998: A framework for integrated services operation over diffserv networks (2000)
16. Sarkar, U., Ramakrishnan, S., Sarkar, D.: Study of long-duration MPEG-trace segmentation methods for developing frame-size-based traffic models. In: Computer Networks. Volume 44. (2004)
17. Wu, M., Joyce, R.A., Wong, H., Guan, L., Kung, S.: Dynamic Resource Allocation via Video Content and Short-Term Traffic Statistics. IEEE Transactions on Multimedia **3** (2001)
18. Mashat, A., Kar, M.: Performance Evaluation of a Scene-based Model for VBR MPEG Traffic. Performance Evaluation IFIP WG7.3 **36** (1999)
19. Haverkort, B.R., Marie, R., Rubino, G., Trivedi, K.: Performability Modelling. Techniques and Tools. John Wiley & Sons (2001)
20. Vallejos, R., Barria, M.: Evaluation of Moments of Cumulative Reward in Reparaible Systems (2005) Submitted to the PERFORMANCE 2005 Conference, Jean le Pins, France.

Design and Implementation of Performance Policies for SMS Systems

Alberto Gonzalez Prieto and Rolf Stadler

KTH Royal Institute of Technology, Sweden
{gonzalez, stadler}@imit.kth.se

Abstract. We present a design for policy-based performance management of SMS Systems. The design takes as input the operator's performance goals, which are expressed as policies that can be adjusted at run-time. In our specific design, an SMS administrator can specify the maximum delay for a message and the maximum percentage of messages that can be postponed during periods of congestion. The system attempts to maximize the overall throughput while adhering to the performance policies. It does so by periodically solving a linear optimization problem that takes as input the policies and traffic statistics and computes a new configuration. We show that the computational cost for solving this problem is low, even for large system configurations. We have evaluated the design through extensive simulations in various scenarios. It has proved effective in achieving the administrator's performance goals and fast in adapting to changing network conditions. A prototype has been developed on a commercial SMS platform, which proves the validity of our design.

1 Introduction

The Short Message Service (SMS) is based on out-of-band message delivery, which permits subscribers to send and receive text messages to/from their mobile phones. SMS was introduced in 1992 and, since then, has experienced a remarkable success: 45 billion messages are sent per month [1], which makes SMS to represent about 10% of the revenue of mobile operators [4].

Controlling a SMS system's performance, especially during congestion periods, is a key management task. This work focuses on a design for performance management of SMS systems that (i) dynamically reconfigures, following the manager's policy, a messaging gateway in response to load changes and network conditions, and (ii) allows a manager to dynamically change management policies if needed.

We apply our design to one specific SMS component: the SMS Gateway (SMSG). The SMSG is a key functional block in the SMS architecture. It is responsible for routing messages between different networks and domains.

We take a policy-based approach to performance management for two main reasons. First, the use of policies permits us to *raise the level of abstraction of the interaction* with the managed device[2][9][10]. This is particularly relevant due to the lack of specialists in SMSGs. Second, policies can be used to specify the operation of *automated management systems*. In this paper, we aim at automating a system that controls the performance of an SMSG.

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 169 – 180, 2005.

We consider both single-class, as well as multi-class SMS services. A multi-class service provides different performance guarantees to different customers or applications. While service providers currently offer only a single class of service, they are considering the introduction of service differentiation. The rationale for service differentiation comes from new uses of SMS messaging such as emergency alarms and promotional messages, which differ in performance requirements. For instance, alarms require low delays, while promotional messages tolerate higher delays or even losses.

The design in this paper supports two classes of SMS services. The first is the priority service; it guarantees delivery with a maximum delay on the gateway. The second is the non-priority service. Messages using this service may be postponed during congestion: they are stored and will be forwarded when congestion is over. The design in this paper dynamically reconfigures an SMS gateway to provide maximum throughput, while observing the quality of service objectives of maximum delays and maximum percentage of postponed messages for the above classes.

Adapting this design to a single-class or more than two classes is straightforward.

The paper extends our previous work on SMS management [17][18] as follows. First, an earlier design has been extended to support additional quality of service parameters, such as the maximum delay. Second, we studied the computational cost of policy re-evaluation. Third, we present results on the trade-off between postponed messages and system throughput. Fourth, we benchmark the performance of our design against that of an ideal system. Finally, we include our experience with implementing our design on a commercial SMS gateway.

The rest of the paper is organized as follows: section 2 discusses performance policies; section 3 describes the SMS architecture; section 4 describes our design for performance management; in section 5, we evaluate our design through simulations in different scenarios; section 6 discusses our prototype implementation; section 7 presents related work; section 8 contains the conclusions the paper.

2 Performance Policies

Administrators want to specify their performance goals in form of *performance policies*. In general, performance policies are derived from business objectives and SLAs. Such policies include performance goals in form of metrical bounds and utility functions that must be maximized. In the general case, performance policies assume a multi-class service system.

Some examples of performance policies are: (i) maximize the number of processed messages, (ii) maximize the number of served customers, (iii) provide low delays for premium customers, (iv) limit the number of postponed messages for customer A to X%, and (v) provide a minimum throughput to customer B of Y messages/second.

Performance policies are given as input to a management system, which maps them into executable functions to achieve the administrator's goals [10][11].

The design for a management system presented in this paper considers an SMS gateway that supports two service classes, priority and non-priority. It supports the following policies: (1) *Maximize the system's throughput* in messages per second.

(2) *Limit the postponement of non-priority messages* to a configurable maximum percentage. (3) *Limit the maximum delay of priority messages* to a configurable value.

Note that an alternative to postponing a message is to drop it. This is not an option for emergency alarms. However, this might be an attractive solution for handling promotional messaging. Our design supports both alternatives and the results we present hold for both of them.

3 SMS Architecture

Figure 1 positions our work within the network architecture for SMS deployment in the GSM context.

The SMSC acts as a store-and-forward system for short messages. Upon receiving an SMS, it queries the HLR database to get the location of the addressee of the message. With this information, the SMSC determines the servicing base station for the addressee and delivers the message to the terminal of the receiver. The SMSC receives messages from two different parties: mobile terminals and SMS gateways.

The SMS Gateway (SMSG) is the functional block in the SMS architecture that interconnects the wireless network to others, such as other mobile operator's network or TCP/IP networks. The gateway's administrator agrees to a traffic profile with the operators of its neighboring SMSCs/SMSGs, typically in the form of a maximum rate.

We use a model for an SMSG that is similar to an IP router. It consists of incoming ports, a routing engine and outgoing ports. Incoming ports receive the messages the gateway has to deliver. On reception, the message is routed to the appropriate outgoing port. After that, the message may need to be converted to a protocol understood by the receiving network. This conversion phase is not considered in this work.

Each outgoing port of the gateway has an associated queue. This permits the gateway to cope with brief periods of congestion. However, longer periods of congestion require control mechanisms.

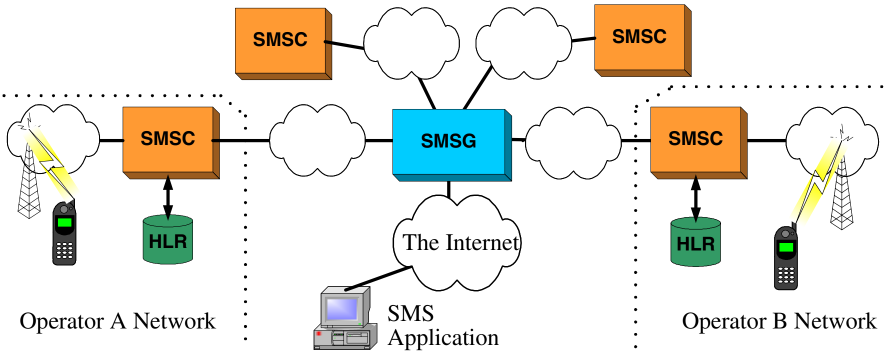


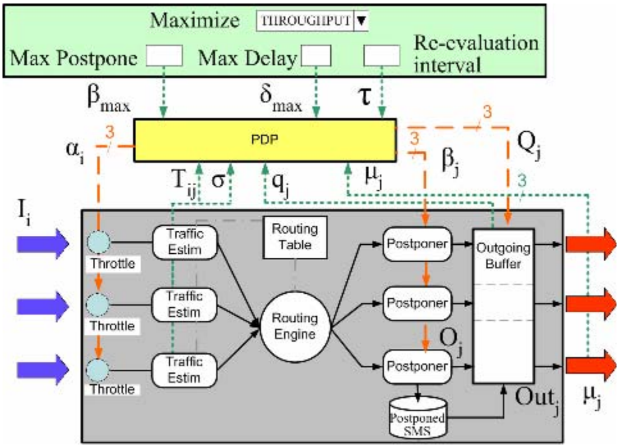
Fig. 1. Functional Architecture for SMS. This work focuses on the SMS Gateway Performance Management**.**

The SMSCs and SMSGs form an overlay network on top of an IP network. The overlay links are created on top of TCP connections. Therefore, ports in a SMSG are software ports, not hardware ones.

A typical port configuration for a large operator has a small number of ports (< 10) with message rates in the order of some tens messages per second. For small operators, configurations often consist of a large number of ports (~20) and lower message rates (< 10 msg/sec).

4 System Design

Functional Architecture for Performance Management. Figure 2 presents the functional architecture of our design. It permits the SMS system to achieve the administrator's performance goals, while adapting dynamically to changes in the load pattern.



max	Maximum percentage of non-priority messages that can be postponed
δ_{max}	Maximum allowable delay for priority messages
α_i	Acceptance rate at port i
j	Percentage of postponed non-priority messages on outgoing port j
I_i	Traffic offered to port i
T_{ij}	Traffic matrix
O_j	Traffic rate sent to outgoing port j
Out_j	Traffic rate sent by outgoing port j
μ_j	Service rate of port j
σ	Percentage of priority messages
q_j	Occupancy of outgoing port j queue
Q_j	Capacity of outgoing port j queue
τ	Policy Reevaluation interval

Fig. 2. Functional Architecture for SMS Gateway Management. The management interface is on top. The PDP block is responsible for the dynamic configuration of the gateway. The bottom block represents the gateway.

The *throttles* in the incoming ports are responsible for limiting the acceptance rates. The *traffic estimators* estimate (i) the traffic matrix, and (ii) the percentage of priority messages. These estimations are used for re-computing the gateway configuration. The *routing engine* decides the outgoing port for each message. It takes this decision based on the information stored in the *routing table*. The *postponer* is responsible for postponing messages, if needed.

The administrator specifies her performance policies for the gateway through the management interface depicted on the upper part of figure 2. In this paper, we consider the policy of maximizing the overall throughput, while observing a maximum

Maximize:

$$\sum_j O_j \qquad \text{(Eq. 1)}$$

Subject to:

$$\sum_i \alpha_i T_{ij}(1-(1-\sigma)\beta_{\max}) \le \mu_j \qquad \forall \text{j} \quad \text{(Eq. 2)} \qquad\qquad O_j \le \sum_i \alpha_i T_{ij} \qquad \forall \text{j} \quad \text{(Eq. 4)}$$

$$\alpha_{\max} \ge I_i \ge \alpha_i \ge 0 \qquad \forall \text{i} \quad \text{(Eq. 3)} \qquad\qquad O_j \le \mu_j \qquad \forall \text{j} \quad \text{(Eq. 5)}$$

The decision variables are α_i.

The values for β_j are determined by:

$$\beta_j = \max\left(\frac{\sum_i \alpha_i T_{ij} - \mu_j}{(1-\sigma)\sum_i \alpha_i T_{ij}}, 0 \right) \qquad \forall \text{j} \quad \text{(Eq. 6)}$$

Fig. 3. Optimization Problem to determine the gateway configuration

delay in seconds for priority messages ($\delta_{\max}$) and a maximum percentage of non-priority messages that can be postponed ($_{\max}$).

We use three mechanisms to achieve the administrator's performance goals for the SMS system. The first controls the *acceptance rate* in the incoming port. In our design, the acceptance rate can be set per port. Note that reducing the acceptance rate on a specific incoming port results in reducing the load on all outgoing ports. The specific values depend on the traffic matrix.

The second mechanism *postpones***,** if needed, some of the non-priority messages routed to congested ports. This mechanism permits having a higher overall throughput at the cost of postponing messages.

These two mechanisms allow the system's administrator to control a trade-off: achieving high system throughput vs. postponing a low percentage of messages. $_{\max}$ is the management parameter that controls this trade-off. It defines the maximum percentage of non-priority messages that can be postponed and takes values from 0% (no messages postponed) to 100% (highest throughput).

The third mechanism sets the *buffer capacity* in the outgoing ports, which controls the maximum queuing delay. In practice, the queuing delay dominates the overall delay of a message passing through the gateway (up to several seconds during congestion periods), while other sources of delay are comparatively small (well below one second).

The PDP (Policy Decision Point). This is the main block of the architecture: it is responsible for dynamically adapting the gateway configuration to achieve the performance goals. The PDP evaluates the performance policies and periodically re-calculates the optimal configuration for the gateway.

The PDP calculates the values for α_i (acceptance rates) and β_j (fraction of non-priority postponed messages). This computation maximizes, for the steady state, the overall throughput, while keeping the postponed non-priority messages below $_{\max}$ and the maximum delay for priority messages below $\delta_{\max}$.

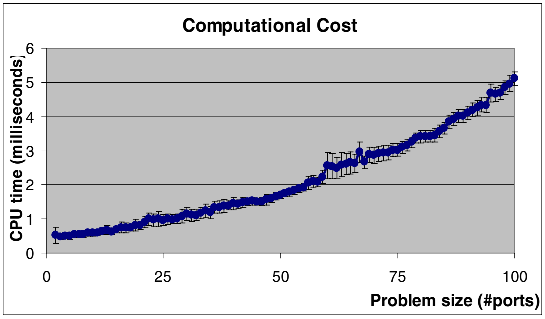


Fig. 4. Computation Cost of the Simplex Algorithm to re-compute the gateway configuration on the PDP

This computation re-evaluates the performance policies every τ seconds as follows. If the occupancy of any queue j is larger that δ_{max} * μ_j, which means that the maximum delay policy is being broken, then α_i will be set to 0 for all i's. Otherwise, the PDP predicts for each queue the future occupancy after the next τ seconds, based on the traffic estimates. If any queue is expected to overflow, then the PDP computes a new gateway configuration by solving the linear optimization problem discussed below. Otherwise, if no queue is expected to overflow, the gateway will be configured to α_i=α_{max} for all i's and β_j=0 for all j's. This means that incoming traffic will be accepted at the maximum rate and no messages will be postponed.

Figure 3 shows the optimization problem that the PDP solves. The objective function (eq. 1) refers to the overall throughput that is to be maximized. The first constraint (eq. 2) states that traffic sent to an outgoing port is limited by the port's service rate. The second constraint (eq. 3) indicates that an incoming port cannot receive more traffic than what it is offered. Equation 6 implies that $\beta_j \leq \beta_{max}$. For a detailed discussion, see [16]. This problem is solved using the well-known Simplex algorithm [5]. Simplex will always find the global solution for all instances of our problem [16].

We have evaluated the computational cost of the Simplex algorithm for re-computing the gateway configuration on the PDP in function of the problem size, which is the number of incoming and outgoing ports. The PDP has been written in C++ and uses the COIN library implementation of the Simplex algorithm [6]. The experiments have been run on an Intel Pentium 1.6 Ghz with 512 MB of RAM with Windows XP Professional 2002 and Cygwin [7]. For a detailed description of the experiments see [16].

Figure 4 shows the results of our evaluation. As expected, the execution times increase with the problem size. However, the algorithm is very efficient in computational terms: it determines the global maximum within a few milliseconds of CPU time, permitting hundreds of policy evaluations per second, even for large configurations. We conclude that performance wise, our design is feasible to realize using current technology.

5 Evaluation Through Simulation

We have evaluated our design through extensive simulation. For this purpose, we have developed a simulator for an SMS gateway that allows us to exercise our design. For details on the simulator implementation, see [16].

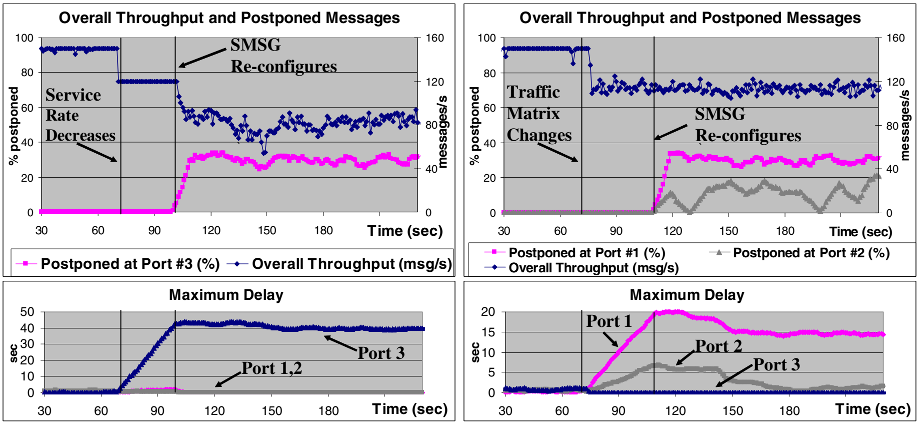


Fig. 5. Evaluation through Simulation. (Left) Service Rate Decrease: μ_3=20msg/sec, β_{max} = 30%, δ_{max} = 50sec, uniform traffic matrix. (Right) Traffic Matrix Change: β_{max} = 30%, δ_{max} = 20sec, non-uniform traffic matrix.

We present simulation results for two scenarios: *service rate decrease in one port*, and *traffic matrix change*. The analysis for a third scenario (*service rate increase in one* port) can be found in [16]. The scenarios share the following characteristics: (i) the port configuration of the gateway consists of three incoming and three outgoing ports; (ii) under normal conditions, the service rate of each outgoing port is 50 msg/sec, and the acceptance rate of each incoming port is 50 msg/sec. The chosen port configuration and rates correspond to a typical configuration for a large operator. (iii) 10% of the messages use the priority service; (iv) the PDP re-calculates the optimal configuration every ten seconds.

The offered load used in the scenarios is based on traces from a commercial SMSG. These traces exhibit low average message rates. To simulate scenarios that are representative for large operator scenarios, we superimposed several of those traces to achieve an average offered load of 50 msg/sec.

We used three different traffic matrices in our experiments. For the first one, each incoming port distributes its traffic roughly evenly among the outgoing ports, each outgoing port receiving about one third. We call this matrix a *uniform matrix*. The second matrix is a *non-uniform matrix*, where the traffic of an incoming port is split unevenly among the outgoing ports. None of these matrices causes congestion under normal conditions. For the third matrix, the *congested matrix*, each incoming port, sends 50% of its traffic to outgoing port 1, 40% to port 2, and 10% to port 3. This causes congestion in outgoing ports 1 and 2.

In the following descriptions, throughput figures refer to 1-second averages; statistics on postponed messages refer to 10-seconds averages; maximum delays are instant values.

Service Rate Decrease in Port 3. In this scenario, we analyze the behavior of our system when the service rate of outgoing port 3 slows down. The experiment starts with all outgoing ports serving at 50 msg/sec. At time=70 sec, outgoing port 3 slows down to 20 msg/sec following a step function, which causes congestion, since the

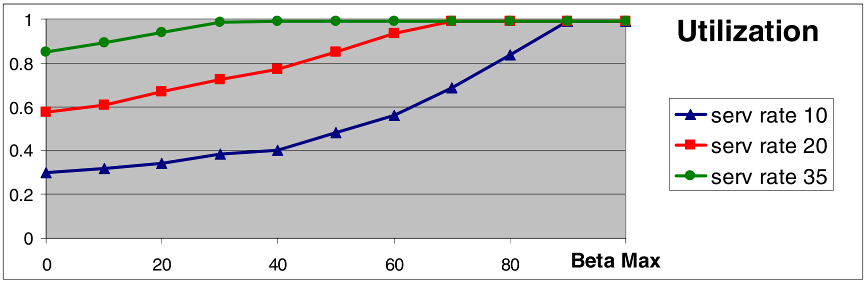


Fig. 6. Controlling the Tradeoff between Achieving Higher Overall Throughput vs. Postponing Fewer Messages

offered load to this port is higher than its service rate. For this experiment, β_{max}=30% and δ_{max}=50 secs. The traffic matrix is uniform.

Figure 5 (left) shows that the system re-configures the gateway 30 seconds after the service rate decreases. This reaction time depends on δ_{max}, μ_j, and the traffic matrix. The dependency on δ_{max} is linear. Higher values of δ_{max} permit the system to cope with longer congestion periods without throttling or postponing messages.

After the system re-configures the gateway, there is a short *transient period* of about 12 seconds. We consider the system stable again when all the outgoing queues are empty, except those of the congested ports. The time to empty the queues depends on β_{max} and on the traffic matrix. It is longer for higher values of β_{max}, since more traffic can be sent to the outgoing ports.

In this scenario, the average throughput in steady state is 87 msg/sec, and 29% of the non-priority messages are postponed.

Traffic Matrix Change. In this scenario, we analyze the behavior of our system, in reaction to a change of the traffic matrix. All outgoing ports serve at 50 msg/sec. The experiment starts with the non-uniform traffic matrix. At this stage, there is no congestion in any outgoing port. At time=70 secs, the traffic matrix changes to the *congestion matrix* following a step function. This change causes congestion in outgoing ports 1 and 2. For this experiment, β_{max}=30% and δ_{max}=20 seconds.

Figure 5 (right) shows that the system re-configures 40 seconds after the traffic matrix changes.

Note that the gateway re-configuration has a marginal effect on the overall throughput. The reason is that only limited throttling is applied, and the postponement mechanism copes with the congestion almost entirely. In outgoing port 1, postponing β_{max} of the non-priority messages reduces the traffic so that it is slightly higher than the service rate. In outgoing port 2, the values for β_2 are well below β_{max}, which is enough to address congestion in this port.

In this scenario, the average throughput in steady state is 113 msg/sec, and 31% (8%) of the non-priority messages are postponed in ports 1 (port 2).

Minimizing Postponed Messages vs Maximizing Throughput. As previously stated, β_{max} controls the trade-off between (i) minimizing the number of postponed non-priority messages and (ii) maximizing the overall throughput. Higher values of

β_{max} allow the system to reach higher throughputs. Lower values of β_{max} result in lower throughputs. Next, we analyze this trade-off.

We have run a number of ‘service rate decrease’ experiments. Figure 6 shows the system utilization (overall throughput divided by the sum of the service rates) as a function of β_{max}. Each line in the graph represents a different service rate for port 3. The statistics we present for each experiment are 140-second averages in steady state.

Our results show that the throughput in steady state depends on β_{max}. This dependency is not linear. The derivative of this function increases with β_{max}, until the system is fully utilized.

The throughput also depends on the traffic matrix. The experiments included in [16] show that the throughput is higher in the case of the non-uniform matrix than for the uniform matrix. This is because the non-uniform matrix permits the system to discriminate better among the sources of messages routed to the congested port. In other words, incoming ports with a small traffic contribution to the congested ports do not need to reduce their acceptance rates significantly.

Benchmarking Against an Ideal System. An ideal system always achieves the administrator’s performance goals for a given policy re-evaluation interval: it (i) keeps the maximum delay at δ_{max}, (ii) never postpones more than β_{max} non-priority messages in a given re-evaluation interval, (iii) ensures that the average traffic sent to an outgoing port never exceeds its service rate in a given re-evaluation interval, and (iv) always achieves the maximum overall throughput.

In contrast to an ideal system, which has complete knowledge of the traffic statistics at any time, a real system or our simulated system has to estimate or predict them. Since the estimation/prediction process is prone to errors, a real system generally performs worse than the ideal one. In our case, the system sometimes breaks the performance constraints. As a consequence, the obtained throughput can occasionally be higher than in the ideal system. For the same achieved performance constraints, the throughput of an ideal system is higher than that of our system.

We compared our design with the ideal system with respect to the overall throughput, the rate of postponed messages, and the maximum delay in steady state. For doing this, we have run a number of ‘service rate decrease’ experiments. Each of them has a different combination of β_{max} and service rate values. The statistics we present for each experiment are 140-second averages in steady state.

In all the experiments we conducted with our design, the constraint on *maximum delay* has never been broken. We explain this by the fact that, when the arrival of a new message in the output queue would break the δ_{max} constraint, a non-priority message from this buffer is postponed.

Our experiments (included in [16]) show that the simulated system tends to slightly outperform the ideal one in terms of *overall throughput*. This is possible since our system occasionally breaks the constraints, caused by inaccurate predictions of the traffic matrix. For most of our experiments, the performance of our design is within 1.5 % of that of the ideal system.

Our experiments (included in [16]) show that the system tends to slightly break the constraint on *postponed non-priority messages*. In most cases, it is not more than 2% above β_{max}. The reasons for this are inaccurate predictions of the traffic matrix and burstiness of the offered load. For a more in-depth discussion, see [16].

6 Prototype Implementation

A prototype of our architecture has been implemented on a commercial SMSG: the Enterprise Messaging Gateway (EMG), version 3.0 [3]. The prototype runs on an Intel Pentium 850 Mhz with 384 Mb of RAM with Linux 2.4 (Debian).

Next, we present our experience with the prototype implementation. Specifically, we discuss how it differs from the simulated model.

In the prototype, the acceptance rate is enforced by controlling the TCP connection with the sender. The SMSG rejects or closes TCP connections to enforce the acceptance rates in each port.

In the simulated model of the gateway, the acceptance and service rates are enforced for control intervals of 0.01 seconds. In a real system, the control interval is generally larger, permitting small bursts of messages. The effect of such bursts is outlined in [16].

In the current version of the prototype, the traffic estimator is the processing bottleneck of the system. The implementation of the traffic estimator is based on analyzing the routing logs generated by the EMG and stored in a database (mysql 4.0 [8]) on the same machine. Currently, it takes between two to four seconds to retrieve and process the data required by the estimators. While the performance of this block limits the time between policy evaluations, an effective re-evaluation period of ten seconds can be achieved.

The reconfiguration of the EMG is not instantaneous as assumed in the simulations. It takes about one second for the EMG to read the new configuration and to apply it.

7 Related Work

Performance and congestion management in routing engines has been extensively studied in the context of IP routers [15]. Our work differs from that work in both the problem space and the solution space. First, congestion management for IP routers considers physical networks. In contrast, an SMSG is a node in an overlay network, where the service rate of outgoing ports can vary, depending on the state of (i) neighboring SMS systems and (ii) the links that connect them. The overlay links are created on top of TCP-IP networks. Therefore, the links' performance is that of a TCP connection.

Second, the approaches to congestion management in IP networks often focus on per-flow end-to-end feedback. Flow-based mechanisms are not relevant in the SMS context, since an SMS message fits into a single packet. In addition, currently, it is not possible to provide congestion-related feedback to the SMS sources. Therefore, such mechanisms are not applicable directly. They would require a major change in the SMS architecture, which is unlikely in the short or medium term.

Congestion control has also been studied by the ATM community. Two main lines were studied [12]. One of the lines was *rate-based control*, which is based on end-to-end control mechanisms. Such approaches are limited by the lack of support to end-to-end feedback.

The other line was *credit-based control* [13], which is based on link by link backpressure mechanisms, as our design is. However, there are two main differences with respect to our work. First, it makes use of per virtual-circuit (VC) control. This allows reducing selectively the rates of the VCs that traverse the congested port, without affecting others. This is not possible for us due to the lack of flows or VCs. TCP congestion mechanisms as RED [14] also benefit from selective reductions of TCP-flows rates. A second difference is a consequence of having per-VC control. These approaches aim at avoiding losses and do not consider postponing/dropping packets.

8 Discussion

In this paper, we presented a policy-based design for congestion management of SMS systems. The design has been evaluated through extensive simulation studies, out of which we described in detail two scenarios: service rate decrease and traffic matrix change.

The results from our experiments are that the system performs remarkably close the administrator's performance goals. First, the overall throughput is within 1.5 % of that of the ideal system. Second, the maximum delay constraint for priority messages is always met. Third, while the system has a tendency to postpone slightly more messages than the given objective, the achieved rate is (in absolute terms) 2% above the given upper bound in most experiments. In addition, our experiments show that the system adapts fast to variations in service rate and traffic matrix.

The simulation studies in [16] suggest that the system is not very sensitive to the traffic characteristics of the offered load. In [16], we present the results for the same scenarios shown in this paper, but using Poisson sources instead of SMS traces. In both cases, the measured performance values, in terms of throughput, postponement rates and delays, are within a few percentage points. We explain this by the fact that the incoming traffic is shaped by the throttles in the incoming ports.

We showed that the computational cost of the policy evaluation, which is performed periodically, is low, even for large system configurations. Policy evaluations can be run on standard microprocessors in the order of milliseconds.

The prototype demonstrates the feasibility of implementing our design on a commercial platform.

Our design facilitates the management of messaging gateways. Compared to today's practices, where administrators often manipulate individual message queues, our design raises the level of abstraction in that an administrator specifies performance goals, and the system adapts its configuration to network conditions. This permits any administrator with a basic understanding of performance metrics to control a gateway, without the need for detailed knowledge of the device internals.

In this paper, we have considered two classes of SMS services. The adaptation of our design to a single-class or more than two classes is straightforward.

We have studied a specific objective function, the overall throughput. Extending our design to alternative objective functions involves the modification of the PDP.

Acknowledgments

We thank Roberto Cosenza from Infoflex Connect AB for his information about SMS systems and their management in commercial environments and for implementing the prototype. We also thank Ulf Brännlund from KTH for his advice on the optimization problem in this paper. This work has been supported in part by VINNOVA under the project Policy-Based Network Management.

References

[1] S. Coulombe and G. Grassel, "Multimedia Adaptation for the Multimedia Messaging Service", IEEE Communications, Vol. 42, No.7, July 2004

[2] M. J. Masullo, S. B. *Calo, "Policy management: an architecture and approach"*. Proc. of IEEE Workshop on Sys. Management, UCLA, Cal., April 1993

[3] Nordic Messaging, www.nordicmessaging.se, August 2005

[4] GSM Association, www.gsmworld.com, May 2005

[5] G.B. Dantzig, "Maximization of linear function of variables subject to linear inequalities", in T.C. Koopmans, editor, "Activity Analysis of Production and Allocation", pages 339-347, 1951

[6] Computational Infrastructure for Operations Reseach, http://www.coin-or.org/index.html, July 2005

[7] Cygwin, http://cygwin.com, August 2005

[8] MySQL, http://www.mysql.com, May 2005

[9] A. Polirakis, R.Boutaba, "The Meta-Policy Information Base", IEEE Network, special issue on Policy-Based Networks, Vol.16, No. 2, pp. 40-48 2002

[10] D. Verma, "Simplifying Network Administration Using Policy-Based Management", IEEE Network, special issue on Policy-Based Networks, Vol.16, No. 2, pp. 20-26 2002

[11] J. Moffett, M. Sloman, "Policy Hierarchies for Distributed Systems Management", IEEE Journal on Selected Areas in Communications, Vol.11, No. 9, pp 1404-1414, Dec 1993

[12] D. Cavendish, M. Gerla, S. Mascolo, "A Control Theoretic Approach to Congestion Control in Packet Networks", IEEE/ACM Transactions on Networking, Vol. 12, No. 5, October 2004

[13] H.T. Kung, R. Morris, "Credit-Based Flow Control for ATM Networks", IEEE Network Magazine, pp. 40-48, March-April 1995.

[14] Floyd S., Jacobson, V., "Random Early Detection gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, Vol.1 No.4, pp. 397-413, August 1993.

[15] A. Mankin, K. Ramakrishnan, "RFC 1254- Gateway Congestion Control Survey"

[16] A. Gonzalez Prieto, R.Stadler, "Policy-based Performance Management for SMS gateways", Technical Report, KTH Royal Institute of Technology, August 2005

[17] A. Gonzalez Prieto, R.Stadler, "Evaluating a Congestion Management Architecture for SMS Gateways", 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005), Nice, France, May 15-19, 2005

[18] A. Gonzalez Prieto, R. Cosenza, and R. Stadler, "Policy-based Congestion Management for an SMS Gateway", IEEE 5th International Workshop on Policies for Distributed Systems and Networks (POLICY 2004), Yorktown Heights, New York, June 7-9, 2004

Detection and Diagnosis of Inter-AS Routing Anomalies by Cooperative Intelligent Agents

Osamu Akashi[1], Atsushi Terauchi[1], Kensuke Fukuda[1], Toshio Hirotsu[2],
Mitsuru Maruyama[1], and Toshiharu Sugawara[3]

[1] NTT Network Innovation Labs., 3-9-11 Musashino-shi, Tokyo 180-8585, Japan
{akashi, terauchi, fukuda, mitsuru}@core.ecl.net
[2] Toyohashi University of Technology, Aichi, Japan
hirotsu@ics.tut.ac.jp
[3] NTT Communication Science Labs., Kyoto, Japan
sugawara@core.ecl.net

Abstract. Verifying whether the routing information originating from an AS is being correctly distributed throughout the Internet is important for stable inter-AS routing operation. However, the global behavior of routing information is difficult to understand because it changes spatially and temporally. Thus, rapid detection of inter-AS routing failures and diagnosis of their causes are also difficult. We have developed a multi-agent-based diagnostic system, ENCORE, to cope with these problems, and improved its functions (ENCORE-2) through our experience in applying the system to commercial ISPs. Cooperative actions among ENCORE-2 agents provide efficient methods for collecting, integrating, and analyzing routing information observed in multiple ASes to detect and diagnose anomalies that human operators have difficulty in handling. ENCORE-2 is also applied to the hijacked route problem, which is one of recent major inter-AS issues.

1 Introduction

The Internet currently consists of more than 15000 ASes (autonomous systems), and this number is still increasing. Inter-AS routing controlled by BGP-4 [1], however, is not stable [2]. Various analyses of this routing behavior and causes of routing instability have been reported [3]. An essential problem is the difficulty of understanding the spread of routing information advertised by an AS [4]. Unlike intra-AS anomalies, the causes of inter-AS anomalies typically exist outside network operator's domain, while the effects of anomalies are sometimes observed only in the advertising AS. This situation is illustrated in Fig. 1. AS_{self} can see the routing information advertised by AS_x and forward packets to AS_x accordingly. On the other hand, packets directed to AS_{self} from AS_x are forwarded according to the routing information advertised from AS_{self}. In this case, AS_{self} has difficulty determining whether its routing information has reached AS_x or was discarded at an intermediate AS where some filter was applied. The operators of an intermediate AS have difficulty detecting this anomaly because

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 181–192, 2005.

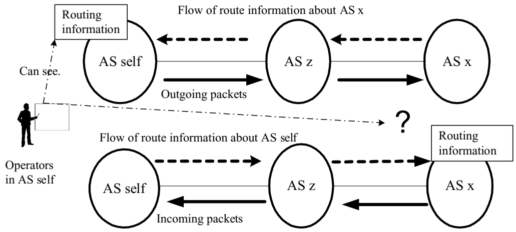


Fig. 1. Problem of verifying spread of routing information

incoming and outgoing packets concerning the intermediate AS are not affected by the filter. Thus, information from other observation points is needed to diagnose this kind of anomaly.

The difficulty of inter-AS routing management comes from noncentralized and autonomous operations. ASes dynamically change their routing relationships with respect to each other. Such temporal changes require on-demand verification and thus invalidate any analysis done in advance. For example, declarative data about each AS's relationships with neighboring ASes are stored in the Internet Routing Registry (IRR) [5], but these data do not necessarily reflect the current statuses of all ASes [6]. Therefore, the data cannot be directly applicable to reachability verification. Moreover, accurate detection of anomalies requires statistical analysis to extract local trends from continuously observed routing information at each observation point. The statistical data, such as the average number or range of BGP full-routes, are required to isolate an anomalous state from a normal one with greater probability. These data are also used to determine when to invoke diagnostic actions.

Centralized analytical approaches [3], some of which use BGP update data collected from multiple ASes, have been proposed, but the autonomy and dynamics of the Internet make performing their analysis difficult and all possible cases are not covered. A method using a cooperative distributed solution (CDS) coincides with this control structure and supplements these other analytical methods. In addition to handling the autonomy of each AS, a CDS would have several advantages over the centralized system approach. From the viewpoint of diagnostic systems, a CDS can be efficient and scalable because 1) statistical calculation to extract local trends in traffic or routing information is performed at the observation points; and 2) the distributed entities, *agents*, exchange only abstracted, analyzed results, rather than raw data. A simple repeated query-and-reply scheme produces a lot of traffic. From the viewpoint of diagnostic functions, a CDS offers higher availability because an agent can act even under the condition where some paths on certain IP networks are unreachable. Agents can try to communicate with each other by relaying messages through a number of cooperative agents. Moreover, a CDS can perform effective analysis because an agent can request other agents to invoke various sensing tools, such as `traceroute` or `ping`, to obtain the remote data and accurately isolate causes of problems. Centralized approaches are, however, incapable of performing these actions at remote points.

To achieve rapid detection and real-time diagnosis of inter-AS routing anomalies, we first analyzed a few years of BGP-related troubleshooting records in our AS to determine the basic functions required for an inter-AS diagnostic system. Then, we designed and implemented a multi-agent-based diagnostic system, called ENCORE [4], which has been applied in actual networks including those of major commercial ISPs for several years and is currently used in commercial operation [7]. The next generation of ENCORE (ENCORE-2) has now developed based on this experience, in order to adapt to recent changes in inter-AS problems. A previous paper [4] described ENCORE's basic diagnostic model and agent architecture, and gave some application examples. This paper focuses on the diagnostic functions of ENCORE-2, which have been extended based on our experience: data collection by agents at multiple observation points, finding indications of anomalies, and analyzing their causes, including the problem of how to handle the hijacked route problem, which is one of recent major inter-AS issues [8]. Anomalies caused by this kind of advertisement were observed a few times a year and were serious problems for commercial ISPs.

2 Analysis of Inter-AS Anomalies

2.1 Difficulties in Inter-AS Routing Management

The difficulties in understanding inter-AS routing can be summarized as follows.

1. [**Spatial changes**]. The routing information is physically and geographically distributed and may vary depending on the observation points.
2. [**Temporal changes**]. The routing information changes over time.
3. [**Administrative domain**]. Routing is controlled independently by each AS. Any operators in other ASes cannot directly access these routing data.
4. [**Local trend**]. Each observation point has its own local trends in the dynamics of routing information. Information about these trends can be acquired only through actual observation at each point and statistical analysis of the observed data.
5. [**Limitation of human operators**]. Detection and diagnosis require human operators to repeatedly observe and analyze large amounts of routing information, including raw data such as BGP update messages. They also require operators to have sophisticated expertise on where and how to collect and analyze data.

The spatial changes easily lead to inconsistent routing states among several ASes, even though each AS is working consistently with respect to neighbor ASes. Moreover, the ASes experiencing anomalies may be different from those causing the anomalies. Therefore, we need to obtain a global view of routing information to verify whether advertised routing information is spreading as the originating AS intends. The temporal changes make advance analysis invalid. Overcoming this problem requires verification at multiple observation points on an on-demand basis. Operators can use tools such as `ping`, `traceroute`, and `looking glass` [9], but they have to use these tools repeatedly over a long period to confirm their own AS's advertisement and find an anomaly as soon as possible.

Table 1. Categories of BGP-related anomalies

category	rate
R1: Received-policy (local)	19%
R2: Received-others (local)	9%
B: Border-area	15%
A1: Advertised (remote)	42%
A2: Advertised-complicated	15%

2.2 Taxonomy of Anomalies

The results of our analysis of BGP-related troubleshooting records from our AS are summarized in Table 1. 28% of the records, denoted *R1* and *R2*, concern received BGP information. *R1* is the set of anomalies caused by erroneous operations when applying our AS's policy by adjusting the attribute values of received BGP information. *R2* is the set of anomalies whose causes do not directly concern BGP, but concern local errors that indirectly affect BGP control. For example, the loss of reachability to the `next_hop` IP address caused by an IGP configuration error belongs to *R2*. No collaborative analysis with other ASes is required because these two groups of anomalies can be locally analyzed.

The remaining 72% of the records cannot be analyzed without BGP information obtained from outside the AS. These records therefore require inter-AS coordination. The third category *B* involves anomalies that occurred in the area bordering the neighbor ASes. Analysis of anomalies in *B* requires status data about the border area such as the connection status of BGP processes and the IP reachability status in the segment used for BGP peering. A part of the data can be observed from the local AS. Their further analysis, however, often requires information observed from neighboring ASes. The *A1* and *A2* categories of anomalies occurred in remote ASes and have almost the same features. They are distinguished by the types of collaborative actions. *A1* accounts for more than 40% of the records and is the set of anomalies that required confirmation in a simple Q&A fashion between the local AS and remote major transit ASes. These anomalies typically occurred after some modification due to configuration changes or maintenance work. Another 15% of the records, which are categorized in *A2*, can also be handled by inter-AS cooperation, but they require more sophisticated actions to analyze the anomalies than those for *A1*. Such actions would include execution of sensing tools from other ASes after exchanging observation results. In some cases, these actions would require changes in cooperative agents to obtain more suitable observation points.

3 Multi-agent-Based Diagnosis

3.1 Required Cooperative Functions

According to the analysis in section 2, a global view of the current routing information that has spread among different administrative domains is essential for diagnosing inter-AS routing anomalies. Since complete understanding of the global view is impossible, we adopt the use of routing information observed almost simultaneously at multiple ASes. By integrating these observed results, we can infer a part of the global view for the purpose of diagnosis. To achieve these coordinated actions, we have proposed a diagnostic system called ENCORE that adopts a multi-agent architecture and utilizes cooperative actions to resolve problems described in section 2.

The basic idea of this system is the reflector model as illustrated in Fig. 2. The essence of this model is to provide a function by which an agent can request a remote agent to observe routing information about a specific AS, which is usually the AS of the requesting agent. The routing information observed and analyzed by remote agents is sent to the requesting agent. Although the reflector model can provide a cooperative function, this function should be performed on an on-demand basis. Thus, a function that enables an agent to request a remote agent to continuously observe the routing information of a specified AS and to notify the requesting agent when specified events occur is required for efficient verification and rapid detection. For example, if the remote agent finds a change in the origin AS number of the BGP attribute value of a specified IP address, it notifies the requesting agent of this observed change. This function is effective because the remote ASes receiving routing information usually become aware of failures earlier than the originating AS.

Another useful function enables the relay of messages to appropriate agents. The relay function is necessary to cooperatively deliver important messages to destination agents even when direct IP routes for message delivery become unavailable. This function is achieved by having the agents act as application gateways. This function is useful because 1) the system can use paths that are not used in usual IP routing, and these paths can include non-transit ASes; and 2) messages whose source IP addresses have changed can pass misconfigured filters with a high probability. Message loops and a significant increase in the number of copied messages are prevented by utilizing information about the path that a message has traversed and restricting the maximum number of hops over which a message can be delivered. When failures are caused by filter setting errors, which are typical configuration mistakes, exchanging messages at the end-to-end level is sometimes impossible. In the case of Fig. 2, if an intermediate AS filters routing information advertised from AS_{self}, AS_{self} cannot access AS_x to verify reachability. In this situation, AS_{self} can exchange messages with AS_x by having an agent in an intermediate AS relay messages because the source

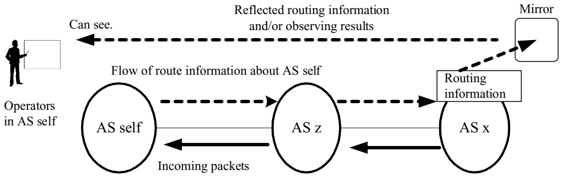


Fig. 2. Reflector model: basic idea for observing spread of information

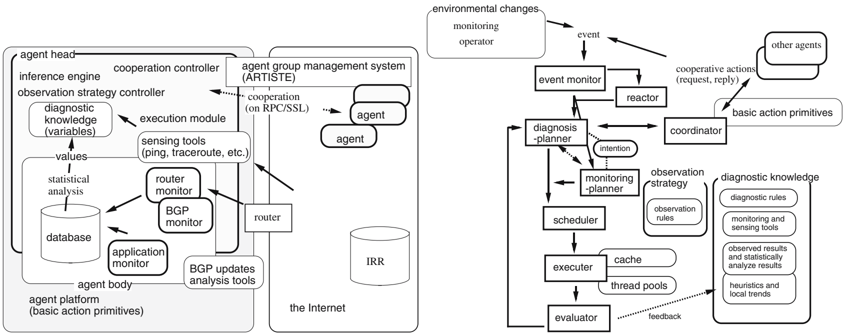


Fig. 3. ENCORE-2 system structure and knowledge processing architecture

IP address of relayed messages changes to another address and this enables the relayed messages to pass the filter.

Each agent needs a strategy that defines how to cooperate with other agents because we cannot assume that agents are located in all ASes in the actual Internet, or agents can act with a large number of agents in all diagnosis phases. For example, the strategy determines a small number of agents that an agent should first access for diagnosis. When an agent starts performing detailed analysis, the agent may need information about other topologically suitable agents. This reorganization requires an on-demand search. Such location information on the BGP topology map is maintained by an agent organization management system called ARTISTE [10], which is an independent system of ENCORE-2. ARTISTE can search agents that match a given requirement, such as “Find agents that can relay messages and are located within 2 AS-hops from ASx”.

3.2 ENCORE System Structure

The ENCORE system was designed based on two assumptions: 1) An ENCORE agent can get the BGP information in the deployed AS; and 2) ENCORE does not require other, special communication facilities. ENCORE-2 consists of several modules classified according to their functions as shown in Fig. 3, and it has been modified and extended based on the design in [4]. The ENCORE-2 agent module is constructed on a network agent platform that provides basic action primitives on distributed environments. They are implemented by using Allegro Common Lisp/CLOS, although the first version of ENCORE was a hybrid system on Gnu Common Lisp, C, and Perl. ENCORE-2 agents use RPC/SSL for authentication and secure communication with each other. ARTISTE is an independent management system that organizes agents located on a BGP topology map. The knowledge processing part of an ENCORE-2 agent, which is based on the BDI (belief, desire, and intention) architecture [11], is also shown. It makes plans for verifying hypotheses, schedules executions of verification rules, and controls monitoring and statistical analysis based on given description. In ENCORE-2, these internal modules works as threads and verification rules are also executed as threads.

3.3 Cooperative Action Management

Agent's roles in the basic cooperative strategy in ENCORE, which are *investigation*, *relay*, and *friend*, are statically assigned to perform required functions for inter-AS diagnosis. ENCORE-2 dynamically searches agents suitable for three roles based on their functional capability and topological conditions. When cooperative diagnosis is performed, an agent sends a query to ARTISTE and it then responds with a list of active agents that can perform the requested role and satisfy a given topological requirement on the BGP map. ENCORE-2 agents can also form groups where role assignments are independently defined. The formation of groups is useful from some political reasons because that can restrict possible cooperative agents and separate management information into each group.

An investigation agent is used to send back requested information observed in its environment. This role is typically assigned to agents located in major transit ASes as in ENCORE, because they can observe the large amount of routing information exchanged there. In ENCORE-2, investigation agents in transit ASes are also used at an early stage of each diagnostic action and are considered as first contact points. In the case these agents would receive a lot of queries from other agents and thus should be able to handle them. Then diagnosis starts and the next investigation agents would be designated for isolating the cause of anomalies in detail and/or identifying an area where that anomaly affects routing. An agent that resides in a stab-AS could be used as an investigation agent although the agent does not have to handle a lot of queries from other agents. A friend agent is utilized to continuously observe the state from outside the AS. In ENCORE-2, candidates for friend agents can be selected using topological requirements such as agents in a neighbor AS, a transit AS, or an AS on the other side of central ASes of the Internet. A relay agent is utilized to control the routing at the application level. If an agent cannot obtain results within a predefined time, the agent selects other investigation or relay agents and requests them to do the job. An initial set of relay agents can also be selected like candidates of friend agents.

An agent may need to find 1) other investigation agents located in ASes downstream from the initially selected investigation agent; or 2) other investigation agents located near the AS in which hijacked routes were observed. These newly selected agents are considered suitable because they could have BGP data to determine the location of the anomaly's cause or the extent to which the anomalous state, such as a hijacked route, is spreading. More comprehensively, ENCORE-2 agents are able to issue search queries to ARTISTE including condition terms such as `group`, `role`, `designated-AS`, `AS-hop-count`, and `topology`, where `topology` is `downstream`, `upstream`, or `neighbor`. Conditions can be combined using logical terms such as `and` and `or`.

4 Diagnostic Knowledge

4.1 Data Acquisition and Local Trends

According to given strategy description, ENCORE-2 statistically analyzes local data and collects analyzed results, if necessary, from multiple ASes. These include actions that are difficult for human operators: 1) Continuous cooperative confirmation of a route advertisement, which requires repeated actions of human operators over a long period. 2) Parameterization of local trends, such as the number and fluctuation of BGP full-route entries, that are also utilized as triggers for starting diagnostic actions. 3) Detailed data analysis using BGP update-level information.

Most of the trends cannot be specified as static values in advance. As an example, one agent in our AS monitors the total number of BGP route entries, which can be an important status indicator. This number differs widely among ASes and changes over time. The total number of BGP route entries in the Internet is currently over 150000 in our AS and can only be acquired through observation. According to our past records, some fatal errors in which many illegal route entries were inserted into a routing table by unintentional advertisement were detected from sudden increases in the total number of BGP entries.

In addition to providing observation functions like a human expert, an agent can provide a more detailed level of monitoring and analysis in terms of frequency and granularity for more accurate diagnostic capability. For example, an agent can monitor BGP update messages [1], while human operators usually see only a part of snapshots of the BGP routing tables in border routers. By monitoring the messages at a lower layer than what humans usually observe, the agent can recognize faults more effectively, as described in subsection 4.3. Consider the case of illegal route insertion. An agent monitoring BGP updates can detect the sudden increase in the number of update messages and easily determine that they originate from the same AS, even if unintentionally advertised routes just overwrite the legal routes and the total number of BGP entries is almost the same.

4.2 Action Strategy

Each ENCORE-2 agent has given action strategies both for the observation and diagnosis phases, which are described based on each AS's policy. For example, an agent R_{self} in AS_{self} may require a friend agent R_x to observe BGP entries and notify it of target IP addresses in AS_{self} and trap conditions. A typical trap condition is "Notify R_{self} if the origin AS number in target IP address entries is changed or some of these BGP entries disappear." When R_{self} is notified that the origin AS number is changed in AS_x, R_{self} schedules possible hypotheses for verification, which include a hypothesis that some routes are hijacked. R_{self} then gets a list of investigation agents from ARTISTE and sends queries about suspicious routes to these agents as shown in Fig. 4. R_{self} can infer the area more completely by repeatedly inquiring of investigation agents near, upstream, or downstream from ASes where unintentional advertisement is detected. The addresses of incrementally required investigation agents are also obtained from ARTISTE. In such partially hijacked cases, relay agents could effectively work to deliver messages among agents. Although serious failures like one in 1997 that disturbed all of the Internet by unintentional advertisement of full-routes might not happen because of careful filters in several major ISPs, partial or small-scale

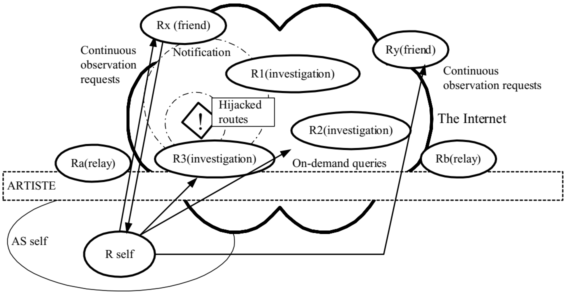


Fig. 4. Cooperative actions for analyzing hijacked route anomalies

unintentional advertisement was observed several times in past few years. Thus, continuous observation in multiple ASes by friend agents and diagnosis using multiple investigation agents is yet to be useful. The former is for rapid detection and the latter is utilized to find out the area where unintentional advertisement affects reachability.

Another scenario is as follows. ENCORE-2 agent R_{self} in stub-AS AS_{self} observes various network status parameters to find indications of anomalies. In this example, suppose a border router in a transit AS, which is located upstream from AS_{self}, fails, and previous router configurations, in which a newly connected AS_{self} is not described, are restored. Then, the advertisement from AS_{self} is filtered. R_{self} finds that it cannot access some hosts, because a rule utilizing `ping` periodically fails. At the same time, friend agent R_n, which observes AS_{self} from the viewpoint of AS_n, can also find these routing changes and try to send a notify message to R_{self}. If this leads to a timeout, R_n then uses relay agents. R_{self} starts diagnosis and tries to send requests to investigation agents, which are R_x, R_y, and R_z. If direct IP forwarding from AS_x, AS_y, and AS_z to AS_{self} is impossible, this step also leads to a timeout. Then, R_{self} asks R_a and R_b to relay the previous request to AS_x, AS_y, and AS_z. Reply messages via R_a and R_b can also be delivered because R_a and R_b reside in the ASes that are not affected by the filter. According to these results, R_{self} generates the next diagnostic plans.

4.3 BGP Update-Level Analysis

For a hijacked route problem, all IP addresses and their AS number in a specific AS can be registered in ENCORE-2 for verifying that the advertised routes from the AS are not hijacked. This verification is achieved by comparing the registered information and observed results in remote ASes. In contrast, for verifying validity of routes from other ASes, the diagnostic system also has to analyze temporal sequences to accurately detect the problem, because the IRR does not necessarily reflect the current status, and the verification method using automatic comparison of the IRR data is not suited for this purpose.

The ENCORE-2 system can act as a BGP peer to monitor and record BGP update-level messages, periodically inserting a short statistical summary. This enables more detailed analysis than snapshot-based data acquisition from a routing table. In this configuration, the short summary contains the times and numbers of update messages, withdrawn routes, and advertised routes per minute. These values are used to efficiently extract the periods in which large numbers of route entries are changed. In the case of the illegal route advertising from AS_z in June 2003, we found illegal update messages from AS_z in a huge number of records. 1) We extracted update records by specifying duration using the `from` time and `to` time, which included the period when we observed some routing failures. In this example, a sufficiently large duration was used to include this period, namely 4 hours. 2) We extracted the ASes that appeared in these records more than 1000 times. At this step, there were four ASes, but AS_z appeared as the origin AS 30 times more often than the other ASes. Thus, AS_z was identified as the origin of the illegal route insertion.

4.4 Applicable Class and Limitations

From the definitions of the basic cooperative functions, this diagnostic model using a multi-agent architecture can be applied to analyze the class of anomalies whose effects can be observed as non-oscillating changes in routing information from outside an AS. According to our analysis, this class covers more than 90% of BGP-related anomalies. On the other hand, there are two types of anomaly records that cannot be analyzed in the current framework. These records belong to categories *B* and *A2*. One type was caused by oscillating changes in routing information both in the local AS and outside the AS through unintended interaction between BGP and an IGP. The second was caused by illegal interaction between the transport layer and the application layer. A hardware failure in a router prevented the router from forwarding IP packets including a specific bit sequence in the data parts. This led to repeated TCP retransmission, and these TCP sessions failed due to timeout errors. Although the latter case can be managed by adding a special heuristic rule, the number of possible hypotheses would increase significantly.

As described above, when an agent finds an anomaly of a given class and tries to diagnose it, the agent must be able to access at least one investigation agent. The model thus inherently requires access to outside the AS, meaning that it cannot cope with anomalies in which some ASes become completely isolated or inaccessible from the Internet. On the other hand, this limitation could be resolved by extending capability of relay agents to use another communication line. Note that unchanged BGP information among multiple observation points outside the AS does not necessarily verify routing validity. There are some cases in which confirmation using only BGP information is insufficient. For example, the IP address designated by the `next_hop`, which is one of the BGP attributes, should be reachable in an AS by using some IGP. If the `next_hop` address is unreachable in the AS, IP packets cannot be forwarded there even if the BGP information is delivered beyond the AS and observed at multiple ASes. In this case, the results of `traceroute` from the outer ASes, which can be performed on the CDS, should be examined as described in a diagnostic rule to reduce the number of possible hypotheses.

5 Related Work

There are several diagnostic tools for analyzing inter-AS routing anomalies. WWW-based systems such as looking glass [9], RIS tools [12], RouteViews [13], and various visualization tools are widely used by network operators to monitor routing information and the states at specific points. These systems, however, are designed to be used by humans, and cannot be straightforwardly applied. Although analysis of temporal and topological behavior of BGP path changes [14] and centralized analysis approaches [3] were reported, all possible cases are not covered. As real-time anomaly detection by analyzing BGP updates, signature-based method and statistics-based method were proposed [15]. These methods can effectively identify anomalous BGP events, but they also cannot cover all cases. Our analysis approach about BGP update events, which utilizes a kind of learned parameters and human operator's heuristics, is less automatic than these methods, it can complementally work with them. As a hybrid system of human and statistically analyzed results [16] is unique and effective. Although it is a kind of visualization tools and cannot be directly applied, it could be complementally work if some patterns were extracted as interpretable rules. Listen and Whisper [17] can eliminate large number of problems due to misconfiguration considering network topology, but Listen only treats verification in the data plane. Whisper can verify routes in the control plane, but it requires another protocol over BGP.

Several advantages provided by the CDS-based approach would be effective to supplement them. From the viewpoints of data availability and cooperation among different administrative domains, some agent-based intra-AS diagnostic systems have been proposed, but these systems only offer restricted cooperation to obtain targeted information. These systems operate under the assumption that the targeted information exists in the management domain of the agent recognizing a problem. This means that the agents in these systems cannot deal with situations in which an anomaly or its effect is observed in a different management domain from that in which the cause exists. This situation is actually quite common in inter-AS diagnosis.

6 Conclusion

To support autonomous and stable operation in the Internet, we have proposed an inter-AS cooperative diagnostic system called ENCORE-2, which is extended through deployment in some commercial ISPs. By using ENCORE-2, an AS can continuously verify that routing is being performed as intended, and can rapidly detect and diagnose a certain class of inter-AS routing failures, which include recent major inter-AS issues such as a hijacked route problem. This CDS approach can effectively supplement other analytical methods through each agent's autonomous actions and cooperation among distributed agents considering the BGP topology.

References

1. Rekhter, Y., Li, T.: "A Border Gateway Protocol 4 (BGP-4)" (1995) RFC1771.
2. The North American Network Operators' Group: (NANOG mailing list) http://www.nanog.org.
3. Feldmann, A., Maennel, O., Mao, Z., Berger, A., Maggs, B.: "Locating Internet Routing Instability". In: Proc. of SIGCOMM, ACM (2004) 205–218
4. Akashi, O., Sugawara, T., Murakami, K., Maruyama, M., Koyanagi, K.: "Agent System for Inter-AS Routing Error Diagnosis". IEEE Internet Computing **6** (2002) 78 – 82
5. Internet Routing Registry: (http://www.irr.net/)
6. Nagahashi, K., Esaki, H., Murai, J.: "BGP Integrity Check for the Conflict Origin AS Prefix in the Inter-domain Routing". In: Symposium on Applications and the Internet, IEEE/IPJS (2003) 276–282
7. : (http://www.ntt.com/release_e/news04/0002/0226.html)
8. Mahajan, R., Wetherall, D., Anderson, T.: "Understanding BGP Misconfiguration". In: Proc. of SIGCOMM, ACM (2002) 3–16
9. Kern, E.: (http://nitrous.digex.net)
10. Terauchi, A., Akashi, O., Maruyama, M., Fukuda, K., Sugawara, T., Hirotsu, T., Kurihara, S.: "ARTISTE: An Agent Organization Management System for Multiagent Systems". In: 8th Pacific Rim Int'l Workshop on Multi-Agents (PRIMA)(To be appeared). (2005)
11. O'Hare, G.M.P., Jennings, N.R.: "Foundations of Distributed Artificial Intelligence". Wiley-Interscience (1996)
12. RIPE: (http://www.ripe.net/)
13. Meyer, D.: (http://www.routeviews.org)
14. Chang, D., Govindan, R., Heidemann, J.: "The Temporal and Topological Characteristics of BGP Path Changes". In: Proc. of Int'l Conf. on Network Protocols, IEEE (2003) 190–199
15. Zhang, K., Yen, A., Zhao, X., Massey, D., Wu, S., Zhnag, L.: "On Detection of Anomalous Routing Dynamics in BGP". In: Proc. of Networking, IFIP (2004) 259–270
16. Teoh, S., Ma, K., Wu, S., Massey, D., Zhao, X., D.Pei, Wang, L., Zhang, L., Bush, R.: "Visual-Based Anomaly Detection for BGP Origin AS Change (OASC) Events". In: Proc. of 14th IFIP/IEEE DSOM. (2003) 155–168 LNCS2867.
17. Subramanian, L., Roth, V., Stoica, I., Shenker, S., Katz, R.: "Listen and Whisper: Security Mechanisms for BGP". In: Proc. of Networked Systems Design and Implementation, USENIX (2004) 127–140

Discovery of BGP MPLS VPNs

Sarit Mukherjee, Tejas Naik, and Sampath Rangarajan

Center for Networking Research,
Bell Labs, Holmdel, NJ
sarit@bell-labs.com,
{tnaik, sampath}@research.bell-labs.com

Abstract. BGP/MPLS VPN is a mechanism defined in IETF RFC 2547 that allows service providers to use their IP backbone to provide VPN services. This mechanism is based on using BGP to distribute VPN routing information to the routers in the backbone network and using MPLS to forward VPN traffic. MPLS tunnels are created dynamically when needed, which relieves service providers of pre-provisioning tens of thousands of tunnels. BGP/MPLS VPNs allow service providers to define any arbitrary topology with any number of nodes in a VPN. The service provider can create multiple VPNs using the same core network. Currently most of the service providers track 2547 VPNs either manually or by using a provisioning database. Algorithms described in this paper aims at automating this VPN discovery procedure. Using our algorithms service providers can automatically discover VPNs that have already been configured using the current network configuration information.

Keywords: BGP, MPLS, VPN, Discovery, 2547, Route Target, VRF, Topology.

1 Introduction

BGP MPLS VPNs as defined in IETF RFC 2547 [1] provide the capability for service providers to use their IP backbone to provide VPN services to their customers. BGP is used to distribute VPN routing information to the routers in the backbone network and MPLS is used to forward VPN traffic. MPLS tunnels are created dynamically when needed. BGP/MPLS VPNs allow service providers to define any arbitrary topology with any number of nodes in a VPN. The service provider can create multiple VPNs using the same core network. Within the context of RFC 2547, a customer site (or more specifically a customer router referred to as a CE router) is connected to the service provider network (or more specifically an edge router on the provider's core network referred to as the PE router) by one or more ports [2][3]. In Figure 1, customer router CE-A is connected to provider edge router PE-A through one port and customer router CE-C is connected to the same provider's edge router PE-A through another port. Thus, multiple CEs could be connected to the same PE as shown in the figure. Within the core network, provider routers (or P routers) function as MPLS Label Switch Routers when forwarding VPN traffic between PE routers.

CE and PE routers exchange routing information using static routing, RIPv2, OSPF or EBGP. A CE router advertises the customer site's local VPN routes to the PE router and learns remote VPN routes from the PE router. After learning local VPN routes

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 193 – 205, 2005.

from CE routers, a PE router exchanges this VPN routing information with other PE routers using IBGP. The service provider associates each of the incoming ports at a PE router to a VPN routing and forwarding (VRF) table. This table contains VPN routing information exchanged by the PE router with the CE router connected to that port. In Figure 1, PE-A has two VRFs, VRF-A that contains VPN routing and forwarding information exchanged with CE-A and VRF-C that contains information exchanged with CE-C. A BGP extended community attribute called the Route Target (RT) attribute identifies a collection of VRFs to which a PE router distributes routes. A PE router uses this attribute to export local routes to other VRFs and to constrain the import of remote routes into its own VRFs. For example, in Figure 1 assume that VRF-A exports a RT and VRF-B on PE-B imports this RT This means, the CE router (CE-B) corresponding to VRF-B knows how to reach hosts behind the CE router (CE-A) corresponding to VRF-A. In order for CE-A to reach hosts behind CE-B, VRF-B needs to export a RT and VRF-A needs to import this RT as well. Once this is done, bi-directional traffic can flow between hosts behind CE-A and hosts behind CE-B. This means, a bi-directional VPN link is established between VRF-A and VRF-B. Thus, the VRFs together with the RTs define the topology of VPNs. In the rest of the paper, when we refer to traffic flow between VRFs we indeed are referring to traffic flow between the CEs connected to the ports on the PE routers on which these VRFs are defined.

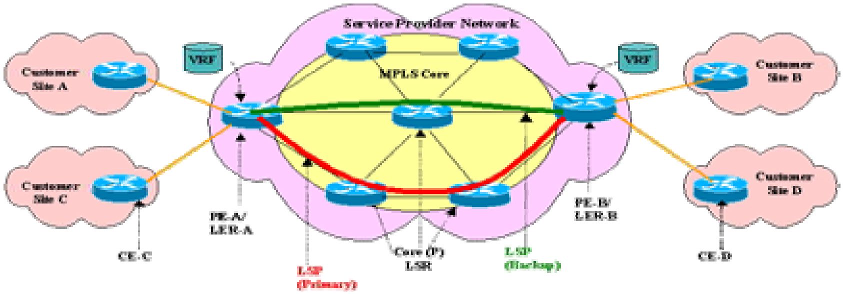
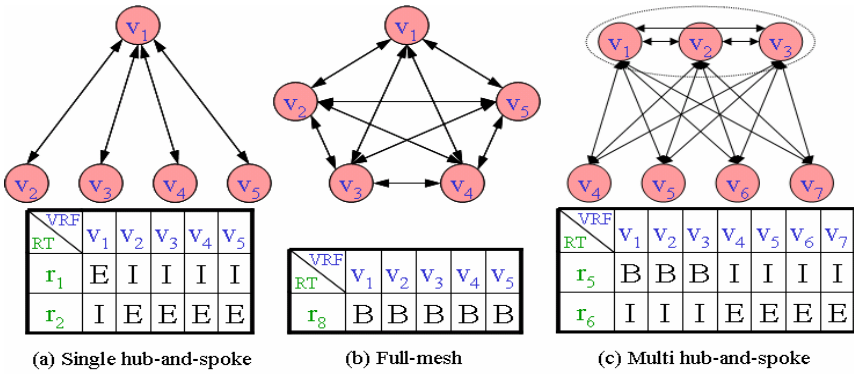


Fig. 1. Example of a BGP MPLS VPN

As illustrated above, a VPN topology can be provisioned using RTs and the export and import of these RTs by the VRFs. Different VPN topologies can be provisioned [2]. Some of the topologies that are normally provisioned include:

- **Single-hub-and-spoke:** In this topology, a single hub VRF can send and receive VPN traffic to a set of spoke VRFs who are not capable of exchanging VPN traffic with each other.
- **Full mesh:** In this topology, a set of VRFs can all exchange VPN traffic with each other. That is, the VRFs are completely connected.
- **Multi-hub-and-spoke:** In this topology, a set of hub VRFs can exchangeVPN traffic among each other as well exchange VPN traffic with a set of spoke VRFs. The spoke VRFs cannot exchange VPN traffic with each other.



VRF / RT	v_1	v_2	v_3	v_4	v_5
r_1	E	I	I	I	I
r_2	I	E	E	E	E

(a) Single hub-and-spoke

VRF / RT	v_1	v_2	v_3	v_4	v_5
r_8	B	B	B	B	B

(b) Full-mesh

VRF / RT	v_1	v_2	v_3	v_4	v_5	v_6	v_7
r_5	B	B	B	I	I	I	I
r_6	I	I	I	E	E	E	E

(c) Multi hub-and-spoke

Fig. 2. Different Atomic and Molecular VPN Components

Figure 2 illustrates these topologies. Figure 2(a) shows a single-hub-and-spoke topology where VRF v_1 is the hub and VRFs v_2,v_3,v_4 and v_5 are spokes. Figure 2(b) is a full-mesh where all VRFs can exchange VPN traffic with each other and Figure 2(c) is a multi-hub-and-spoke where VRFs v_1,v_2 and v_3 form a full-mesh and a multi-hub and the spokes are v_4,v_5,v_6 and v_7.

2 Related Work

One important issue that arises after the VPNs are configured is the problem of tracking the configurations of already existing VPNs using the network configuration information. Although tools are available that provide the ability to configure VPNs, locate physical faults that may occur within the VPNs and measure MPLS VPN traffic and quality of service [4][5], to the best of our knowledge, we are not aware of tools that, given the current VRF and RT relationship information, automatically generate the different components that make up the VPN. Such a discovery tool would provide the necessary information to ascertain a) if the VPNs are configured according to specifications, b) find redundancy in the configurations, and c) provide information to visualize the VPN in terms of component topologies described earlier. In this paper, we present an algorithm to implement such a discovery tool. Our algorithm is not specific to any vendor so it can be applied across any vendor. The IETF draft [6] talks about how one PE can discover other PEs in a VPN using a protocol. Our algorithm discovers global view of all the VPNs using offline algorithm. We implemented mandatory steps of algorithm in our company product.

3 Algorithm for Topology Discovery

When the VRFs are provisioned, they may be provisioned using a minimum number of RTs. For example, to provision a full-mesh, only one RT is needed. As long as a single RT is defined on all the VRFs and is exported and imported by all the VRFs, VPN connectivity is established between every pair of VRFs thus leading to a full-mesh topology. Similarly, to provision a single-hub-and-spoke or a multi-hub-and-spoke only two RTs are needed. One RT will be exported by the (multi) hub which will be imported by all the spokes and all the spokes will import a single RT which will be imported by the (multi) hub. We refer to the largest such components provisioned using the minimum number of RTs as atomic and molecular components as defined below.

Definition 1 [Atomic Component]: The largest single hub-and-spoke with two RTs and the largest full-mesh with one RT are atomic components. Figure 2 (a) and (b) are examples of atomic components.

Definition 2 [Molecular Component]: The largest multi hub-and-spoke with two RTs without any restriction on overlapping links and nodes with atomic components is called a molecular component. Figure 2 (c) is an example of molecular component. Note that it is composed of four atomic components, one full-mesh and three single hub-and-spokes.

In Figure 2, a VRF-RT table is used to represent the export-import relationship between VRFs and RTs. An E entry denotes that the RT represented by the row is being exported by the VRF represented by the column. Similarly, an I entry denotes an import. An entry of B denotes that the RT represented by the row is being both imported and exported by the VRF represented by the column. From the VRF-RT tables above it can be seen that they do represent atomic and molecular components.

An important problem to be solved is the discovery of different components that a VPN is comprised of (that is, the topology of the VPN in terms of different components) and this problem is the focus of this paper. In addition to discovering atomic and molecular components, which are provisioned using minimum number of RTs, we are also interested in discovering basic components such as full-mesh, single-hub-and-spoke and multi-hub-and-spoke even if they are provisioned using more than the minimum number of RTs. In this regard, we define two other types of components.

Definition 3 [Composite Component]: The largest single hub-and-spoke or the largest full-mesh or the largest multi hub-and-spoke components without any restriction on the number of RTs are composite components. Therefore, by definition all atomic and molecular components are composite components.

Definition 4 [Complex VPN]: A complex VPN is a union of composite components.

A composite component is either a single-hub-and-spoke or mult-hub-and-spoke or full-meshes without any regard to the number of RTs that are used. It is clear that each of these components is a union of unique atomic and molecular components (that use the minimum number of RTs). This means, a composite component could be represented as a union of unique atomic and molecular components. By definition, a complex VPN is a union of composite components. Given this definition, it is clear that **a)** the set of atomic components in a complex VPN is unique, and **b)** The set of atomic and molecular components in a complex VPN are unique after the links and nodes belonging to set of atomic components that overlaps with set of molecular components are taken out of the set of atomic components.

The above observation implies that a complex VPN topology can be uniquely represented using a set of atomic and molecular components, whereas representation using composite components may not be unique. The goal of our algorithm is to discover complex VPNs as a composition of atomic and molecular components and in order to do that, we first identify all the atomic components, construct molecular components from the atomic components, if any and then represent a complex VPN as a union of these atomic and molecular components. The optional goal is to represent a complex VPN as a set of composite components.

3.1 Requirements of the Discovery Algorithm

In addition to discovering complex VPNs, we pose some additional requirements on our algorithm that optimizes the discovery process as outlined below.

1. Find all the redundant RTs, that is the RTs that are not producing any new topology which is already not discovered. Denote the set of redundant RTs as $\mathcal{R}$.
2. Find all the unidirectional links. Denote the set of unidirectional links as $\mathcal{U}$.
3. Find all the atomic and molecular components of the complex VPN. Denote the set of atomic full-mesh components as $\mathcal{F}$, atomic single hub-and-spoke as $\mathtt{S}$ and molecular multi hub-and-spoke components as $\mathcal{M}$.
4. Find and express the topology of the complex VPN if it is made of composite components

3.2 The Discovery Algorithm

We assume that route distribution is purely due to BGP MPLS VPN. It is not affected by route redistribution, filter or route maps on PE or CE routers.

Given a description of a VPN (using RTs), it can be decomposed into different sets of components. We use (f_1, f_2, …, f_x) to denote a full-mesh created using nodes f_i, i = 1, …, x. We use ($h \rightarrow s_1$, s_2, …, s_x) to denote a single hub-and-spoke with h as the hub and s_i, i = 1, …, x as the spokes. Similarly we use (h_1, h_2, …, $h_y \rightarrow s_1$, s_2, …, s_x) to denote a multi hub-and-spoke with h_i, i = 1, …,y as the hubs and s_i, i = 1, …, x as the spokes. Note that (h_1, h_2, …, h_y) is a full-mesh. The steps of our algorithm are enumerated below.

Step 1: Prepare VRF-RT Table: For the given network, let the number of VRFs be n and the number of unique RTs be m. Number the VRFs as v_1, v_2, …, v_n and number the RTs as r_1, r_2, …. r_m. Prepare a m×n table, referred to *VR*, where RT r_k, $1 \leq k \leq m$ forms the kth row and VRF v_i, $1 \leq i \leq n$ forms the ith column of the table. Fill in the entries in the *VR* table with B, E or I according to the specified RTs. The preparation of this table would take O(nm) time.

Note: Remove a row from the VRF-RT table if the row has only one B, all E's or all I's.

Above step will remove all RTs that are not capable of forming unidirectional or bidirectional link and hence do not form participate in any VPN. This removes the corresponding RT from the set of RTs.

Step 2: Construct Adjacency Matrix: Construct a graph based on the VRF-RT table as follows [7]. The VRFs are the nodes of the graph. Put a directed edge with label r_k from node v_i to node v_j, $i \neq j$, if RT r_k from VRF-RT table is exported by node v_i and imported by node v_j. Let the edge be represented by $(v_i, v_j)r_k$. Treat B's as both E and I. In an adjacency matrix (*AM*) representation of the graph, create a n×n matrix with $AM(v_i, v_j) = r_k$ if there is an RT r_k, $1 \leq k \leq m$ in the VRF-RT table that is exported by node v_i and imported by node v_j and $i \neq j$; i, j = 1, …, n. Note that the diagonal entries of the matrix are left empty. In Step 1, all redundant RTs would have been removed. This means, a directed edge between two VRFs would exist only due to one RT. Even then, in the worst case, to determine the existence of an edge between two VRFs (say v_i and v_j), all the column entries in the VRF-RT table corresponding to VRF v_i need to be checked. This means, to find all edges between all pairs of nodes would take $O(n^2m)$ time.

Step 3: Determine Unidirectional Links: A link qualifies as unidirectional if the nodes it is directed between do not have another link going in the opposite direction. In the adjacency matrix representation, if $AM(v_i, v_j)$ exists, but $AM(v_j, v_i)$ does not, then (v_i, v_j) is an unidirectional link, $i \neq j$; i, j = 1, …, n. Remove all the unidirectional links from the graph and put them in $\mathcal{U}$. Therefore,

$$\mathcal{U} = \{ (v_i, v_j)r_k \mid AM(v_i, v_j) = r_k \wedge AM(v_j, v_i) = \Phi, i \neq j, 1 \leq i, j \leq n, 1 \leq k \leq m\}.$$

A simple algorithm to determine this would require all entries in the adjacency matrix to be checked once and thus would take $O(n^2)$ time.

Step 4: Reduce Set of RTs: The general RT reduction algorithm is the following:

1. Denote by binary variable x_{ri}, $1 \leq i \leq m$ if RT r_i is present in the set of minimal RTs.
2. Consider each cell in the adjacency matrix. Let $(r_1, r_2, \ldots, r_p)$ represent the set of RTs in that cell. Introduce a constraint as $x_{r1} + x_{r2} + \cdots + x_{rp} > 1$.
3. Minimize $\sum_{i=1}^{m} x_{ri}$ subject to the above set of constraints.
4. Solve the minimization problem and the x_{ri}'s thus achieved give the minimal RT set.

Note that if we want to keep some RT i for some reason then we should make that $x_i = 1$ in the constraint set. If we want to give preference on removal of one RT over another, then the objective function can be changed to

$$\text{Minimize} \sum_{i=1}^{m} w_i \, x_{ri},$$

where w_i is a relative weight on the RT. If we want to remove a RT in preference of another, then the former should be given higher weight.

Remove the rows containing the redundant RTs from the VRF-RT Table. Also remove the redundant RT from the cells in the VRF-VRF Table. For the rest of the algorithm, assume that the set of RTs has been reduced following the algorithm. The number of RTs may have been reduced, and the reduced number is still denoted by m, and there is no gap in the sequence of RTs. The algorithm to reduce the set of RTs to a minimum number of RTs reduces to a node covering problem in a graph and is known to be NP-complete. The discovery algorithm does not require this reduction to work correctly and hence this is an optional step.

Step 5: Determine Set of Atomic Full-Mesh Components: If an RT in VRF-RT table has more than one B's then output the set of nodes with B's as a full-mesh. Put them in $\mathcal{F}$. Now remove the B's from the table. The effect of this is to remove all the corresponding links in the graph, i.e., the entries in *AM*, but not the nodes. We define $b_k = \{v_i \mid VR(r_k, v_i) = B, 1 \le i \le n\}$, which is the set of all the VRFs which both imports and exports r_k (i.e., B in the cell in the VRF-RT table for row r_k). Therefore,

$$\mathcal{F} = \{b_k \mid \ |b_k| > 1,\ 1 \le k \le m\}.$$

This step requires all entries in the VRF-RT table to be checked once in the worst case and thus would require O(nm) time.

Step 6: Create Set of Candidate Hubs: In order to discover all the atomic single hub-and-spoke components, we start with selecting a hub. A node with out-degree one or more qualifies for this[1]. The set is referred to as the set of candidate hubs and is denoted by $\mathcal{CH}$.

$$\mathcal{CH} = \{v_h \mid \exists\, i, k, \text{ s.t. } VR(v_h, v_i) = r_k,\ 1 \le i \le n\ 1 \le k \le m\ \}.$$

A simple implementation of this step which consults, in the worst case all entries in the VRF-VRF adjacency matrix, would require $O(n^2)$ time.

Step 7: Create Set of Preferred Hubs: A node in an atomic full-mesh component may become a hub in a molecular multi hub-and-spoke component. It can happen only if the RT used for determining the atomic full-mesh has I in some its entries. In order to facilitate the determination of molecular components, we prepare a set of preferred hubs denoted as $\mathcal{PH}$. Therefore,

$$\mathcal{PH} = \bigcup_{k=1}^{m} \{f_k \mid (f_k \in \mathrm{F})\ \wedge\ (\exists\, i\ VR(r_k, v_i) = I,\ 1 \le i \le n)\}.$$

In the worst case, the set $\mathcal{F}$ may consist of all n VRFs (that is, the complex VPN is made up of one single atomic full-mesh) and this means all entries in each of the columns of the VRF-RT table may have to be checked. This would require O(nm) time.

Step 8: Determine Set of Atomic Single Hub-and-Spokes: We determine how many of hubs from $\mathcal{CH}$ becomes part of an atomic single hub-and-spoke. In order to qualify, there must be two distinct RTs, one where the candidate hub exports to a set of nodes and the other where the candidate hub imports from the same set of nodes.

While there are elements in $\mathcal{CH}$ do the following:

1. For each node $v_h \in \mathcal{CH}$, find all the distinct RTs r_k, $1 \le k \le m$,used to export from v_h. Form the set of spokes $S(v_h, r_k)$ for each distinct RT r_k.

$$S(v_h, r_k) = \{s \mid VR(v_h, s) = r_k\}.$$

[1] Note that since the unidirectional links are removed, in-degree and out-degree of a node are the same.

2. For each of the set of spokes $S(v_h, r_k)$, $1 \leq k \leq m$, find the largest subset of nodes that uses the same RT to export to the hub. The cardinality of the largest subset is the in-degree of the hub.

$\mathit{In\text{-}Degree}(v_h) = \max_{1 \leq k \leq m} | \{s \mid VR(s, v_h) = r_j, s \in S(v_h, r_k), r_j \neq r_k, 1 \leq j, k \leq m\}|$.

3. Find the hub $v_h \in \mathcal{CH}$ with the largest in-degree. If multiple hubs qualify, then select a hub $v_h \in \mathrm{PH}$. Make $(v_h \rightarrow \{s_i \mid VR(s_i, v_h) = r_j, s_i \in S(v_h, r_k), r_j \neq r_k, 1 \leq j,k \leq m\})$. Include this single hub-and-spoke, $(v_h \rightarrow s_1, \ldots, s_x)$ in S. Therefore,

$$\mathrm{S} = \mathrm{S} \bigcup \{(v_h \rightarrow s_1, \ldots, s_x)\}.$$

4. Remove all links associated with this single hub-and-spoke component $(v_h \rightarrow s_1, \ldots, s_x)$ from the graph. That is assign $AM(v_h, s_i) = AM(s_i, v_h) = \Phi$.
5. Remove singleton nodes (i.e., nodes with no incoming and outgoing links) from $\mathcal{CH}$.

For each of the VRFs, the (sub) step 1 would take O(m) time. To find the in-degree of each of the hubs in (sub) step 2 would take O(m) time in the worst case considering all nodes that form the spoke for the hub. To find the hub with the largest in-degree in (sub) step 3 would require O(n) time in the worst case. Sub steps 4 and 5 take constant amount of time. Thus, for each of the elements in $\mathcal{CH}$ the time spent in the above steps is O(m+n). The set $\mathcal{CH}$ may contain all n VRFs and thus it would take $O(mn + n^2)$ time to execute the above sub steps.

Step 9: Determine Set of Molecular Multi Hub-and-Spokes: The following steps prepare the set $\mathcal{M}$.

1. From the set $\mathcal{F}$, take a new full-mesh component all whose nodes are members of $\mathcal{PH}$. That is, $b_k \in \mathcal{F} \wedge b_k \subseteq \mathcal{PH}$, $1 \leq k \leq m$. Stop if there are no new full-mesh components for consideration.
2. Check if each of the nodes of the full-mesh component b_k is a hub in the set $\mathcal{S}$ of single hub-and-spoke.
3. Check for each atomic single hub-and-spoke (where the hub $\in b_k$) the RT exported by the hubs is the same and the RT exported is the same one for creating full-mesh b_k. Also check that the RT imported by the hubs is the same.
4. If the test passes, put the full-mesh and the associated single hub-and-spoke components into the multi hub-and-spoke set $\mathcal{M}$ and remove them from $\mathcal{F}$ and $\mathcal{S}$, respectively. Otherwise go back to step 1.

Each node in set $\mathcal{F}$ could be such that it is not part of a full-mesh component. In this case, for each of these nodes, sub-step would take O(m) time as each RT may have to be checked. Thus, the time required to execute the above sub-steps would be O(nm).

Steps 8 and 9 are the core of the algorithm where the complex VPN is discovered and represented as a composition of atomic and molecular components. This is the main goal of the algorithm. The steps described below are optional and serve to provide more information about the Complex VPN in terms of composite components.

Step 10: Check if the Complex VPN is a Composite Full-Mesh: We determine if the complex VPN is a composite full-mesh. That is every node is directly reachable from every other node. This is verified from the adjacency matrix if each entry in the upper triangular matrix without the diagonal has a valid RT entry in it. If so, the complex VPN is a composite full-mesh. The adjacency matrix can be checked in O(n^2) time.

Step 11: Check if the Complex VPN is a Composite Single Hub-and-Spoke: We determine if the complex VPN is a composite single hub-and-spoke. It is straight forward by making sure that $\mathcal{F}$ and $\mathcal{M}$ are empty and all the single hub-and-spoke components in $\mathcal{S}$ have the same hub. The time for this step in the worst case is O(n) if all VRFs form the hub for some single-hub-and-spoke component.

Step 12: Check if the Complex VPN is a Composite Multi Hub-and-Spoke: We determine if the complex VPN is a composite multi hub-and-spoke by doing the following:

1. Determine the largest full-mesh component in the graph. This is done by finding the largest square sub-matrix with same set of nodes in the rows and columns from the adjacency matrix such that each entry of the sub-matrix has a valid RT in it, except for the diagonal of the sub-matrix which may or not have any entry. Denote by $\mathcal{CF}$ the set of composite full-mesh which is composed of the nodes of the sub-matrix.
2. From the set $\mathcal{S}$, combine two single hub-and-spokes into one single hub-and-spoke if they both have the same hub. Continue till no more combinations are possible. The set thus formed is called the composite single hub-and-spoke and represented by $\mathcal{CS}$.
3. Check if the set of hubs formed from $\mathcal{CS}$ is same as the set $\mathcal{CF}$. If so check if each single hub-and-spoke component of $\mathcal{CS}$ has the same set of spokes. If the test passes, and $\mathcal{CS}$ contains all the nodes of the network, then the complex VPN is a multi hub-and-spoke topology.

This step requires the largest clique to be identified within a graph [8] as indicated in sub-step 1 and hence is NP-complete [9].

Note that if the RT reduction phase of the algorithm is not conducted, then

- It is possible that after the determination of Set of Molecular Multi Hub-and-Spokes there are still some entries left in the adjacency matrix. The corresponding RTs will be unidirectional and redundant and will not produce any new topology.
- During step 8, remove unidirectional links, if any.

The above algorithm finds all atomic and molecular components and thus the complex VPN that is composed of these components. Ignoring the optional steps (Step 4 and Steps 10, 11 and 12), the complexity of the algorithm is bounded by Step 2 which is the time for constructing the adjacency matrix. Thus the running time of the algorithm is O(n^2 m) where n is the number of VRFs and m is the number of RTs.

We have implemented all the non-optional steps of the algorithm and the algorithm has been used to discover large complex VPNs in terms of atomic and molecular components as part of a Lucent Technologies' network management product. An illustrative example of the steps of the algorithm can be found in the appendix.

4 Open Issues and Future Work

We presented the design of an algorithm for VPN discovery that analyzes complex VPN configurations and represents these VPNs in terms of simple atomic and molecular components. Our algorithm uses a graph model to represent a complex VPN and decomposes this graph to identify the simple components.

The proposed algorithm uses two optional steps, i.e., reduction of set of RTs and determining if the complex VPN is a composite hub-and-spoke, which are NP-complete. For large complex VPNs and open issue is to design efficient approximation algorithms to solve these steps. Another open issue is to extend the algorithm to incrementally discover VPN topology changes. That is, when changes are made in the VRFs of the PE routers, VPN topologies are modified without rerunning the entire algorithm.

In the future, we plan to extend the algorithm to differentiate between intranets and extranets. This involves associating the customer profile information from a provisioning database with the output of our algorithm. In addition we intend to apply the proposed techniques on other types of VPNs that are similar to BGP/MPLS VPNs such as VPLS. Future work also includes integrating the output of our algorithm to BGP MPLS VPN monitoring systems.

References

[1] E. Rosen, Y. Rekhter, "BGP/MPLS VPNs", Internet RFC-2547, available at http://www.ietf.org/rfc.html, March 1999.

[2] P. Tomsu, G. Wieser, "MPLS-Based VPNs Designing Advanced Virtual Networks", Pearson Education, December 2001)

[3] C. Semeria, "RFC 2547bis: BGP MPLS VPN Fundamentals", available at http://www.juniper.net/solutions/literature/white_papers/200012.pdf.

[4] HP Openview Network Services Management Solution for MPLS Networks, Available at http://www.hp.com.

[5] Youngtak Kim, Hyung-Woo Choi, Hyo-Sung Kim, "A QoS-guaranteed DiffServ-aware-MPLS VPN and its Network Management System", SNPD, 2003

[6] Hamid Ould-Brahim, Eric C. Rosen, Yakov Rekhter, "Using BGP as an Auto-Discovery Mechanism for Layer-3 and Layer-2 VPNs", IETF draft, June 2005

[7] Aho, J.Hopcroft, J. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley Longman Inc., April 1978.

[8] Harary, "Graph Theory", Addison Wesley Publishing Company, January 1995.

[9] Garey and D. S. Johnson. Computers and Intractability, "A Guide to the Theory of NP-Completeness", Freeman, 1979

Appendix

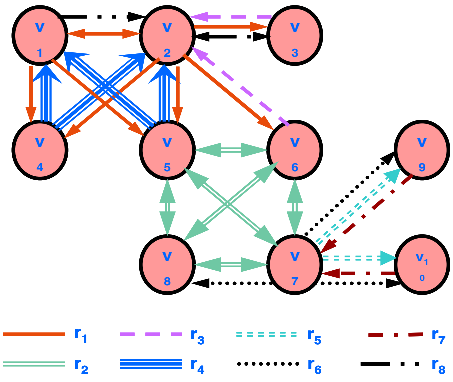
The following example explains important steps of the algorithm to discover all 2547 VPNs using a sample network whose VRF-RT table is shown below. Due to space limitation, we do not elaborate some of the steps.

Step1: Construct VRF-RT Table

RT \ VRF	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
r_1	B	B	I	I		I				
r_2					B	B	B	B		
r_3		I	B			E				
r_4	I	I		E	E					
							E		I	I
r_6							B	I	I	I
r_7							I		E	E
r_8	E	B	B							

Step 2 and 3: Construct adjacency matrix and remove unidirectional links
$\mathcal{U}$ = { $(v_1,v_3)r_{1,}r_8$, $(v_1,v_6)r_1$, $(v_6,v_3)r_3$ }

VRF \ VRF	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
v_1		r_1,r_8	🟥	r_1	r_1	🟥				
v_2	r_1		r_1,r_8	r_1	r_1	r_1				
v_3		r_3,r_8								
v_4	r_4	r_4								
v_5	r_4	r_4				r_2	r_2	r_2		
v_6		r_3	🟥		r_2		r_2	r_2		
v_7					r_2	r_2		r_2,r_6	r_5,r_6	r_5,r_6
v_8					r_2	r_2	r_2			
v_9							r_7			
v_{10}							r_7			

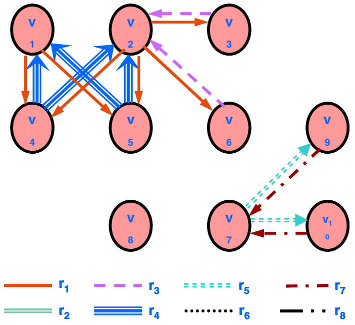


Step 4: Reduce Set of RTs: Minimize $\sum_{i=1}^{8} x_i$ subject to $0 \le x_i \le 1$, $1 \le i \le 8$ and $x_1 \ge 1$, $x_2 \ge 1$, $x_3 \ge 1$, $x_4 \ge 1$, $x_7 \ge 1$; $x_1 + x_8 \ge 1$, $x_2 + x_6 \ge 1$, $x_3 + x_8 \ge 1$, $x_5 + x_6 \ge 1$.

The solution to this problem is $x_1 = x_2 = x_3 = x_4 = x_7 = 1$ and either x_5 or x_6 is 1. We choose $x_5 = 1$. So the reduced set of RTs is $\{r_1, r_2, r_3, r_4, r_5, r_7\}$.

Step 5: Determine Set of Atomic Full-Mesh Components. $\mathcal{F}$ = $\{(v_1,v_2), (v_5,v_6,v_7,v_8)\}$

VRF \ VRF	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
v_1		▦	🟥	r_1	r_1	🟥				
v_2	▦		r_1	r_1	r_1	r_1				
v_3		r_3								
v_4	r_4	r_4								
v_5	r_4	r_4				▦	▦	▦		
v_6		r_3	🟥		▦		▦	▦		
v_7					▦	▦		▦	r_5	r_5
v_8					▦	▦	▦			
v_9							r_7			
v_{10}							r_7			



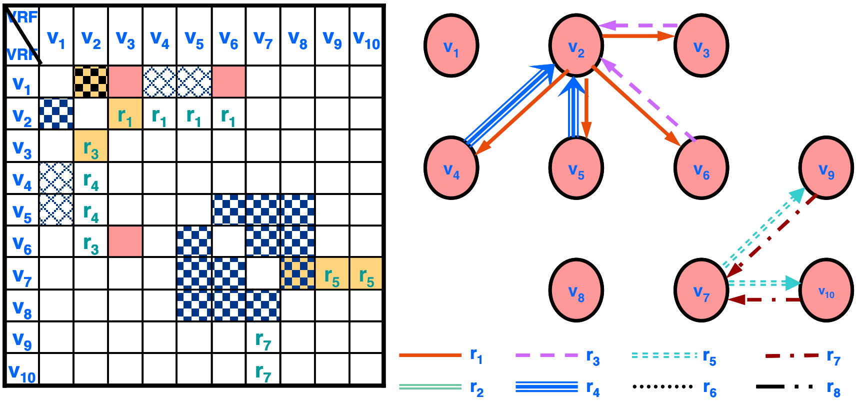
Step 6: Create Set of Candidate Hubs. $\mathcal{CH}$ = {v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_9, v_{10}}.

Step 7: Create Set of Preferred Hubs. $\mathcal{PH}$ = {v_1, v_2}.

Step 8: Determine Set of Atomic Single Hub-and-Spokes. For each element of the candidate hub set, we find the set of spokes reachable using one RT. From the set of spokes we then find subsets of nodes that use the same RT to export to the hub. Then we compute the cardinality of the largest such subset.

Spoke set	Elements	Export to hub	In-Degree
S(v_1, r_1)	{v_4, v_5}	{v_4, v_5} to v_1 using r_4	max |{v_4, v_5}| = 2
S(v_2, r_1)	{v_3, v_4, v_5, v_6}	{v_3, v_6} to v_2 using r_3 and {v_4, v_5} using r_4	max{|{v_3,v_6}|,|{v_4,v_5}|} = 2
S(v_3, r_3)	{v_2}	{v_2} to using r_1	max |{v_2}| = 1
(v_4, r_4)	{v_1, v_2}	{v_1, v_2} to v_4 using r_1	max |{v_1, v_2}| = 2
S(v_5, r_4)	{v_1, v_2}	{v_1, v_2} to v_5 using r_1	max |{v_1, v_2}| = 2
S(v_6, r_3)	{v_2}	{v_2} to v_6 using r_1	max |{v_2}| = 1
S(v_7, r_5)	{v_9, v_{10}}	{v_9, v_{10}} to v_7 using r_7	max |{v_9, v_{10}}| = 2
S(v_9, r_7)	{v_7}	{v_7} to v_9 using r_5	max |{v_7}| = 1
S(v_{10}, r_7)	{v_7}	{v_7} to v_{10} using r_5	max |{v_7}| = 1

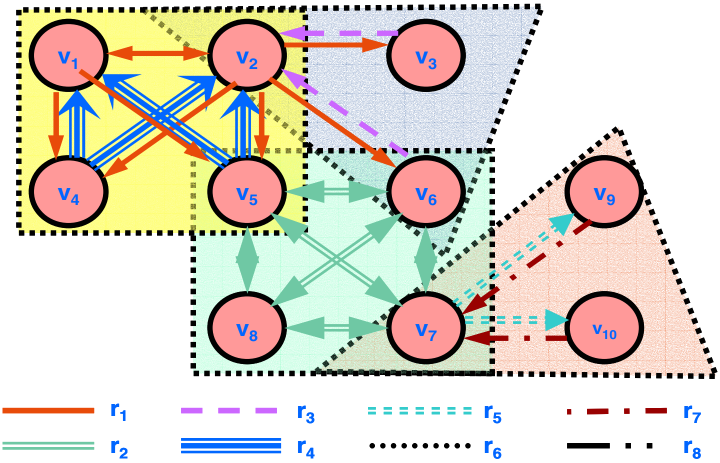
Hubs qualified for selection are v_1, v_2, v_4, v_5, v_7. We select ($v_1 \rightarrow v_4$, v_5) since $v_1 \in$ $\mathcal{PH}$. Therefore, $\mathcal{S}$ = {($v_1 \rightarrow v_4$, v_5)} and now $\mathcal{CH}$ = {v_2, v_3, v_4, v_5, v_6, v_7, v_9, v_{10}}. After removal of this atomic component the graph is as follows.



Similarly we find atomic components ($v_2 \rightarrow v_3$,v_6), ($v_2 \rightarrow v_4$,v_5) and ($v_7 \rightarrow v_9$,v_{10}). Therefore $\mathcal{S}$ = {($v_1 \rightarrow v_4$, v_5), ($v_2 \rightarrow v_3$, v_6), ($v_2 \rightarrow v_4$,v_5),($v_7 \rightarrow v_9$,v_{10})} and now $\mathcal{CH}$ = { }.

Step 9: Determine Set of Molecular Multi Hub-and-Spoke. We see that (v_1,v_2) $\in \mathcal{F}$, and both v_1 and v_2 are hubs in $\mathcal{S}$. We see that ($v_1 \rightarrow v_4$,v_5) and ($v_2 \rightarrow v_4$,v_5) are present in $\mathcal{S}$. RT exported by v_4 and v_5 is r_4 and it is the same as the one imported by v_1 and v_2. Therefore,

$$\mathcal{F} = \{v_5, v_6, v_7, v_8\},\ \mathtt{S} = \{(v_2 \rightarrow v_3,v_6),\ (v_7 \rightarrow v_9,v_{10})\},\ \mathtt{M} = \{(v_1,v_2 \rightarrow v_4,v_5)\}$$



The complex VPN used in this example is neither a composite full mesh, nor a composite single-hub-and-spoke or a composite multi-hub-and-spoke. Thus, steps 10, 11 and 12 do not apply.

Policy-Based Adaptive Routing in Autonomous WSNs

Carlos M.S. Figueiredo[1,2], Aldri L. dos Santos[3], Antonio A.F. Loureiro[1], and José M. Nogueira[1]

[1] Dept. of Computer Science, Federal University of Minas Gerais,
Belo Horizonte-MG, Brazil
{mauricio, loureiro, jmarcos}@dcc.ufmg.br
[2] FUCAPI - Research and Tech. Innovation Center, Manaus-AM, Brazil
[3] Dept. of Computer Science, Federal University of Ceará, Fortaleza-CE, Brazil
aldri@lia.ufc.br

Abstract. Wireless sensor networks (WSNs) are employed in different domains and applications. The resource constraint on such networks, many times composed of hundreds to thousands of devices, and the requirement of autonomous operation become their management a challenging task. This work applies policies, a well-known approach in network management, in the core task of routing in autonomous WSNs. Policies are used to establish rules to take dynamic actions on the network according to its state. Our scheme offers a high-level and flexible way to realize management tasks related to routing in WSNs, which can be defined in a progressive way as knowledge from the environment is acquired or application requirements change. Case studies employing a policy-based adaptive hybrid solution allows the autonomous selection of the best routing strategy in view of network conditions and application requirements. Simulation results show the benefits and resource savings offered by the use of policies for adaptive routing in WSNs.

Keywords: Wireless Sensor Networks, Routing, Policy-based design.

1 Introduction

A wireless sensor network (WSN) consists of sensor nodes connected among themselves by a wireless medium to perform distributed sensing tasks [21,1]. These networks are employed in different applications such as environmental and health monitoring, surveillance, and security. An important aspect of WSNs comes from the fact that many sensors generate sensing data for the same set of events. The integration of sensing, signal processing, and data communication functions allows a WSN to provide a powerful platform for processing data collected from the environment.

WSNs diverge from traditional networks in many aspects due to a large number of nodes with strong energy restrictions and limited computational capacity. In general, WSNs demand self-organizing features, i.e., the ability of

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 206–219, 2005.

autonomously adapt to the changes resulted from external interventions, as topological changes (due to failures, mobility or node inclusion), reaction to a detected event, or requests performed by an external entity.

The objective of such network is to collect and process data from the environment and send it to be further processed and evaluated by an external entity connected to a sink node (or gateway). Consequently, routing towards the sink node is a fundamental task and different algorithms have been proposed to this purpose [2]. However, different applications and scenarios demand algorithms with different features. Thus, given a specific scenario, the WSN can be designed to operate with the most appropriated routing algorithm, which can be defined *a priori*. However, in some cases the variations of these scenarios can be constant or even unpredictable. For example, an event-driven scenario may present a low incidence of events and, at a given moment several events can be detected generating a high traffic. Thus, a WSN should support autonomic solutions appropriated for different periods, since it might be infeasible or undesirable to an external entity to act dynamically on the network in order to adapt its behavior.

Adaptive and hybrid approaches for routing in WSNs consist of viable solutions to deal with variable scenarios (e.g. [23,7,22]); but, they generally provide rigid solutions for some specific cases. Policies are a well-known approach in network management that give a high-level and flexible way to realize management tasks [24]. This work shows how policies can effectively be used in autonomous WSNs for the task of routing. Further, it is illustrated an adaptive hybrid case for the autonomous selection of the better routing strategy taking into account both network conditions and application requirements. Simulation results show that the usage of policies for adaptive routing in WSNs can provide resource savings and benefits.

The rest of the paper is organized as follows. Section 2 extends the discussion on routing in WSNs and the usage of policies found in related proposals. Section 3 considers the conception and implementation of policies on these networks. Section 4 describes case studies of adaptive routing using policies, and the associated results are discussed in Section 5 showing the advantages of this approach. Section 6 presents our final considerations and future work.

2 Routing and Policies

In this section, we briefly discuss the main approaches to routing in WSNs and the usage of policies on networks.

2.1 Routing in WSNs

Routing in WSNs differs from traditional networks in many aspects. Essentially, energy efficiency is the main requirement in such constrained-resource networks and the routing activity should also consider it. Further, the basic goal of a WSN is to collect and process data from the environment and send it to be processed and evaluated by an external entity. Thus, routing towards the sink node is a fundamental task and different algorithms have been proposed [2], each one of them being more suitable for a given case or scenario due to its specific features.

Basically, we found the following classes of protocols with respect to routing infrastructure building for WSNs: *Flooding or Gossiping* are classical mechanisms to forward data in sensor network that do not need to maintain any routing infrastructure or topology [8]. In the flooding mechanism, every node forwards data by broadcasting it to all its neighbors until it reaches the destination. Gossiping differs from flooding by choosing random nodes to forward the data. Although these mechanisms disable the cost of route creation and maintenance, they cause the data packet implosion problem which represents an excessive cost for WSNs. *Proactive protocols* are routing algorithms that create and maintain the routing infrastructure no matter the network behavior. In general, this process is realized by the destination nodes. Examples of proactive protocols are DSDV [20] for MANETs and various tree-based protocols for WSNs such as One-Phase Pull Diffusion [9], SAR [26], and some implementations in [28]. Although this approach can result in a better routing process, it has the disadvantage of a constant resource consumption. *Reactive protocols* are routing algorithms that build the routing infrastructure only when a node wants to transmit a packet, i.e., it is a source-based approach. AODV [19] is a well-known protocol for ad hoc networks and Push Diffusion [9] is an example for WSNs with such behavior. This approach turns resource savings in possible inactivity periods, but it may cause the overhead of path discovery for each source node.

Depending on the application, routing can be performed considering different models [27], such as continuous, event-driven, and observer-initiated. Continuous and observer-initiated models are favorable for proactive protocols, whereas event-driven are adequate for reactive ones. Thus, given a fixed scenario, the WSN can be designed to operate using the most adequate routing algorithm defined *a priori*. But, the application requirements may change as well as may exist scenarios where the behavior of the network varies in an unpredictable way along the time. Such situations aim different (pro-active, reactive) algorithms at different instants, becoming infeasible or undesirable to an external entity to act dynamically on a network to change its behavior. Given these scenarios, we should design routing protocols based on autonomic principles.

Adaptive and hybrid approaches for routing in WSNs are viable solutions to deal with variable scenarios. For example, in [7] we considered an event-driven scenario previously illustrated, where the data traffic is monitored in time intervals and a reactive or proactive behavior is taken if this traffic stays below or above a specified threshold, respectively. In [23], Shen et al. presented a hierarchical architecture with a query language such that different routing strategies can be taken autonomously according to the application requirements and the query nature. However, these solutions treat some specific cases previously considered by the designers. Our intention in this work is to discuss a flexible way to implement an adaptive hybrid routing solution using policies.

2.2 Policies on Networks

Policy-based systems have been studied mainly in management tasks for distributed systems and networks [24]. Policies can be specified as rules governing the choices on the behavior of a system, allowing changes in such system without the requirement of rebulding it. Thus, such systems must support dynamic update of policy rules interpreted by distributed entities to modify their behavior.

Different proposals have been presented in the literature on specification and deployment of policies, such as languages, frameworks and deployment models. Typically, policies are established by low-level specific rules which consider individual configuration parameters. However, high-level abstractions, through the specification of system goals, may turn easier the administrator task. For example, the framework described in [12], which is contextualized in autonomic computing, interrelates three types of policies: Action Policies, which dictate the actions that should be taken whenever the system is in a given current state, typically in the form of "IF(Condition) THEN(Action)" clauses; Goal Policies, which specify either goals for a desired system state or criteria that characterize them, rather than specifying exactly what to do in the current state; and Utility Function Policies, which define an objective function that expresses the value of each possible state, thus providing a more fine-grained and flexible specification of behavior than the Goal Policy.

Action Policies compose the low-level specific rules, and the other policy types are high-level abstractions which must be translated in a sequence of specific rules to be executed by the system components. Clearly, policies can be used in the design of WSNs. They establish a way to formalize the actions that will be taken from local interactions among network elements or from network perceptions. Flexibility is provided with the redefinition capacity of policies. Thus, new local rules can be established due to a new global goal or an unpredictable situation, for example.

Regarding network routing, some studies [4,5] propose the use of policies in order to obtain cost savings and quality of service (QoS) in the routing task. In this work, we deal with specific characteristics and solutions of WSNs.

3 Policy-Based Adaptive Routing in WSNs

We propose the use of policies in the task of adaptive routing in WSNs. The objective is to provide a flexible mechanism to define autonomous routing operation to achieve a better efficiency. Policies must establish rules to dynamically act on WSNs based on analysis of collected data. In the routing context, these actions can trigger changes in the routing strategy, such as an adaptive solution that chooses dynamically either a reactive or a proactive behavior based on the measured traffic [7], or still in the case of a proactive tree [29], change the parent selection strategy according to a specific routing metric.

Generally, existing adaptive routing solutions for WSNs are rigid and do not allow changes in their adaptive rules. The great advantage of a policy-based solution is its possibility to be redefined by the management entity for matching new application requirements, fixing or improving the network performance due to an unpredictable situation.

3.1 A General Model

A generic model for the use of policies in WSNs is shown in Fig. 1. The main idea is to separate basic routing mechanisms, difficult to implement and change, from the strategy of the adaptive process, usually defined by high-level rules, which are easy to build and can lead to better performance. In fact, basic routing mechanisms, or even the entire node, can be reprogrammable to allow the desired flexibility (e.g. dynamic code load [11]). But, such process can be highly complex for a management entity or demand more computacional resources from nodes to update and execute larger and more complex dynamic code.

Nodes that run policies must contain a module called *policy processor*. This module provides to the network nodes the ability to store, interpret, and execute specific policies. It can be a virtual machine that runs a script, the capacity of loading code dinamically or simply the execution of a parameter assignment, depending on how policies are defined and implemented in a given architecture. These policies can interact with the applications and other nodes in order to achieve their goals.

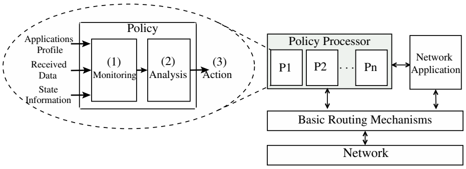


Fig. 1. A generic model for policies in WSNs

Policies must include functionalities to monitor, analyse, and act on a network. The monitoring phase (1) uses both data and state information collected from the network to analyze and use it in future actions. The profile (requirements) of each application running on the network can also be taken into account when making an action decision. The analysis phase (2) may use techniques from data fusion, intelligent agents, such as accounting, comparison with thresholds, prediction or inference techniques, and others to better detect situations where an action is needed (3).

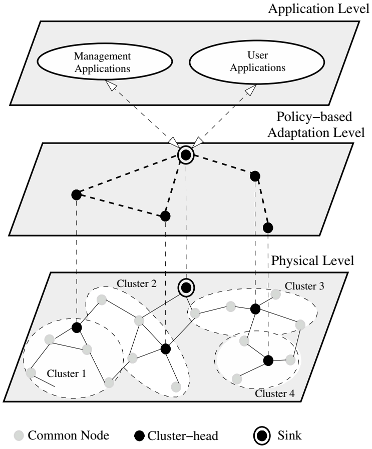


Fig. 2. Hierarchical structure

This policy model can be applied to an individual network node, determining its own routing strategy based on its network perceptions, but it may be very difficult to keep the consistency of the entire network to perform the cooperative data collection task. Thus, this model is more viable only to strategic nodes, as sink nodes, which have more visibility of the network for monitoring and can take a proper autonomous monitoring decision for the whole network.

Hierarchical structures are very common in the network management and, in particular, in WSNs through clustering [10,14,13], allowing the grouping of related nodes for some collaborative task, minimizing the number of message transmissions, providing scalability, and dividing the complexity of the network management in sub-domains. Thus, such organization is also considered in the policy-based routing as depicted in Fig. 2. In the physical level, clusters are composed of common nodes coordinated by a *cluster-head* node. These cluster-heads usually compose an intermediate level where policies can be defined to provide the better routing strategy or parameter optimization in different moments for each cluster. Such policies follow the model previously described and use network perceptions collected from the cluster to choose its routing strategy. In the top level, applications, often external to the network, enable users and management entities to interact with the network, access data collected from clusters, and assign policies to different nodes in the network. The interaction between external entities and nodes happens through the sink node.

This hierarchical scheme for adaptive routing can provide resource savings and benefits. In particular, different parts of large scale networks are subjected to different kinds of queries (determined by the applications), different traffic characteristics and environment influence, which may cause topological changes on network. Thus, the creation of various adaptive subdomains allows the adoption of the best routing strategy for each one of them.

3.2 Implementation

A Basic Routing Mechanism Set. We propose the use of flooding, proactive and reactive routing mechanisms since they enable a WSN to operate in the three forms as described in Section 2.1. *Flooding* is fully infrastructureless and does not require any action from the receiver; thus, all data are propagated by broadcasts. The *proactive* mechanism, used is the classical tree-based approach that enables periodical constructions of a routing tree rooted at the destination node to create paths towards every network node. An implementation where the parent adoption strategy of a node is based on the firstly tree-build message received from its neighbors, called EF-Tree (Earliest-First Tree), is used in this work. The *reactive* mechanism adopted is based on the AODV behavior [19], where the routing process is triggered by source nodes only when a data is available. The routing process starts with a data flooding on the network. When a data message arrives at the destination node, that node will respond with periodical requisition messages to the neighbor node that sent the first data message. Thus, the response is recursively propagated in the same manner towards the data source node, establishing a route and inhibiting new floodings. The SID (Source Initiated Dissemination) protocol [7], which also provides similar behavior, is used in this work.

In the mechanisms described above, we note that the routing infrastructure starts at the destination node, generally the sink node in WSNs. To apply these mechanisms in a transparent way, we propose an integrated forwarding rule to change the routing algorithm based on the destination node decision, which can be defined by a policy. The rule is defined as follows: every data packet (generated or to be forwarded) is sent through a routing tree if it exists and it is valid (EF-Tree behavior); otherwise, the data is sent to the neighbor in which a specific and valid requisition was received (SID behavior); In latter case, data is sent by flooding. All nodes respond to control messages as their original protocols determine, but the validity is ensured by a timestamp and a predefined time period.

With this forwarding rule, anticipated tree-building messages (as EF-Tree) are used to achieve a proactive behavior. Such messages can also carry commands or queries to define data collection parameters (e.g., continuous collection and data rate). In the absence of proactive messages, nodes operate in an event-driven mode. Previously configured parameters characterize the events that need to be notified. When such events occur, the source nodes flood their data on the network. If the receiver desires to create a reactive routing infrastructure, it responds using the SID algorithm. If flooding is preferred to be maintained, the receiver only needs to receive data without any reaction.

Policy Implementation. There are several forms to implement policies in distributed systems and networks (Section 2.2). Scripts are preferable because they are codified in high-level languages, which give more versatility and power to the administrators. Further, there are solutions based on scripts to WSNs, such as the TinyScript of the Maté platform [15], found in the Mica2 architecture [6]. TinyScript is a high-level script language that is compiled to a bytecoded version to be distributed to the sensor nodes and executed in a specific virtual machine. This encoding allows resource savings and simpler interpretation on distribution and execution, which is needed in this kind of network.

```
privatecounter;
counter=getNumNodes();
if counter > THRESHOLD then
  takeAction(PROACTIVE);
else
  takeAction(REACTIVE);
endif
```

(a) Policy script

```
getnn            // read number of nodes
pushc THRESHOLD  // puts THRESHOLD on operand stack
inv              // invert the THRESHOLD
add              // number of Nodes − THRESHOLD
blez 8           // if ((NN − THRESHOLD) > 0)
                 // go to proactive
pushc REACTIVE   // puts REACTIVE on operand stack
act              // take a REACTIVE action
halt
pushc PROACTIVE  // puts PROACTIVE on operand stack
act              // take a PROACTIVE action
halt
```

(b) Bytecode encoding

Fig. 3. Policy implementation

An implementation of the routing rule of Fig. 4, described in Section 4, using Maté's specification is exemplified in Fig. 3. It shows a simplified script, Fig. 3(a), and its bytecoded version, Fig. 3(b), configured to check periodically the number of nodes sending data in a cluster, `getNumNodes()` function, in order to select the routing strategy empsloyed (`PROACTIVE` or `REACTIVE`). If such value is higher or lower than a threshold, a proactive or reactive strategy is assumed, respectively, using `takeAction (`⟨type⟩`)` function. Both functions must be built in the application to be executed on the node together with the Maté virtual machine.

A limitation of the Maté platform is the restrict set of instructions, reducing the policy codification possibilities. More powerfull script approaches exist for WSNs, such as the Sensorware platform [3], however it requires a more complex interpreter in the nodes and increases the number of data bytes transmitted on the network.

Policy Deployment. An advantage of the policy-based adaptive routing is the flexibility to redefine the rules when the network is in operation and based on the knowledge acquisition. Some strategies must be used to deploy them in the nodes. A flooding scheme is a simple solution to reach all nodes, although it may add a significant over-head for WSNs. We expect policy implementations to be small (simple rules) and their redefinitions not a frequent task. But, robust mechanisms are required to keep the network in a consistent state, since a new node or a sleeping one can be (re)joined to the network with an older policy version. Some solutions have proposed energy-efficient and robust code update mechanisms for WSNs, such as Trickle [16], which can be applied in our proposed approach.

4 Case Studies

Next we present two case studies showing the use of policy-based adaptive routing. The first case considers a scenario where different routing strategies are applied autonomously according to a policy definition. The second case deals with an adaptive parameter set instead of a constant default value. In both cases, the goal is to achieve a better performance regarding energy consumption, a constrained resource in WSNs.

4.1 Changing the Routing Strategy

We consider an unpredictable scenario where traffic variations may occur or the application requirements can change along the time. In such cases, different routing strategies may be more appropriated for different moments. Thus, an adaptive rule can be applied to achieve a hybrid behavior by switching among different routing strategies according to the monitored network condition. For example, in an event-driven scenario, the network may remain with a very low activity for days, favoring a reactive algorithm. But, in a given moment, a number of events may occur, generating a traffic large enough to use a proactive algorithm.

For this scenario, we have proposed an adaptive hybrid algorithm which determines the best routing strategy (reactive or proactive) based on the number of sources sending data [7]. This number can be counted observing the different source node identifiers (ids) and storing them in a bit-array to save memory, for example. A threshold used by the sink node evaluates the measured network traffic (number of source nodes sending data). If the traffic is lower than the given threshold, a reactive strategy is adopted; otherwise, a proactive strategy is taken. A rule that represents this routing strategy is shown in Fig. 4.

Although this simple implementation can lead to better performance than the specific algorithms, the traffic measurement does not show the advantage of a proactive action if a threshold is reached but no new detections happen. For example, occasional distribution of event detections can lead to a concentrated measurement, higher than the established threshold, leading to a proactive behavior, which may be not necessary.

As a better adaptive model, we consider the use of the simple signal processing method of Moving Average Filter (MAF) [25] on the number of nodes that are starting to detect events. Thus, a detection is counted observing when a source node starts to send data after its innactivity. The filter smoothes possible measurement noises and captures the seasonal component of event-occurrence on the last n monitoring periods (n is the filter window size). Hence, the filter can give a good estimation for the next time period, helping in the decision to take a proactive behavior or not. Assuming the basic mechanisms described before, if more than one new source is expected for the next period, the proactive behavior can be used to build the routing infrastructure for the entire network with the cost of a route discovery. Fig. 5 shows a routing strategy rule for this solution.

```
if Num_of_Sources > THRESHOLD then
  proactive_action();
else
  reactive_action();
endif
```

Fig. 4. Routing rule 1

```
Next_Est=MovingAverage(Num_of_Detections);
if Next_Est > 1 then
  proactive_action();
else
  reactive_action();
endif
```

Fig. 5. Routing rule 2

```
if Num_of_Sources < old_Num then
  proactive_action();
endif
old_Num = Num_of_Sources;
```

Fig. 6. Optimization rule

4.2 Adaptive Routing Optimization

Often, routing algorithms for WSNs define different parameters that must be configured according to a given application and network situation to achieve a better performance. But if the network condition changes, a policy can detect it and set dynamically the better routing parameter. For example, in routing solutions presented in Section 2.1, it is common the use of periodical updates to support topological changes caused, for instance, by failures. Pre-defined frequent updates can be set to a scenario with common failures, but this configuration is not appropriated to a stable scenario with a low failure rate due to the consequent communication overhead. If it is possible to determine the failure occurrence rate, a policy can dynamically change this routing parameter, or even define when to take the corrective action, which optimizes the routing performance.

In this case study, let us consider that a given application starts requiring continuous environmental measurements. Thus, a proactive routing strategy such as EF-Tree is more adequate. Instead of setting a constant periodical rebuilding rate, in [17] we apply data fusion techniques in a solution that monitors the traffic and infers about failure occurrence to take this action only when necessary to save energy. This failure detection is possible by assuming a continuous traffic, thus, downsteps in the received traffic mean a failure possibility. In this work, we build a simplified similar rule, shown in Fig. 6, in which the number of nodes sending data (counted as described in Rule 1) is observed in the periods of the data generation rate. A reduction of this number means a failure, which triggers a tree building to recover it.

5 Simulation and Evaluation

We evaluated the viability of our policy-based scheme through simulations. Experiments were performed using the ns-2 simulator [18]. The simulation parameters were based on the Mica2 node [6] using the 802.11 protocol in the MAC layer. Through its datasheet, bandwidth is 19200bps, radio range is 40m and Tx. and Rx. power consumption are 45 and 24mW, respectively. In all simulations, we assumed a network size of 50 nodes randomly distributed in an area of 100×100 m^2 with only one sink, and transmission of data and control messages (tree-building or requisition messages) of 20 bytes every 10 s and 100 s, respectively. The graphics show the summary result of 33 simulations and the error bars (vertical bars) represent the confidence interval of 95%. The evaluated metric was the total energy consumed by all sensor nodes. Simulation results consider scenarios and routing rules described in Section 4. For simplicity, we embedded the basic routing mechanisms and the policies at sensor nodes.

5.1 Changing the Routing Strategy

In a non-correlated event-driven scenario, we varied the number of sources generating data randomly, with a uniform distribution, along the simulation time of 4000 s. A policy equivalent to the routing rule of Fig. 4 was set to run in intervals of 10 s, the same period of data messages, and the traffic collected in an n-interval is compared with a static threshold in order to set the strategy activated in the $(n + 1)$-interval. Thus, it is expected that different thresholds lead to different results, as in Fig. 7, where it is shown the behavior of a network running with EF-Tree and SID algorithms alone, which compose the basic routing mechanism set described in Section 3.2, as well as the Policy-Based scheme with traffic thresholds (limit of the number of source nodes sending data to take an adaptive behavior) of 1, 2 and 3, called *PB-1*, *PB-2* and *PB-3*. This scheme autonomously adapts the routing mechanisms during the simulation, as described before. A threshold higher than 3 leads to a performance very close to

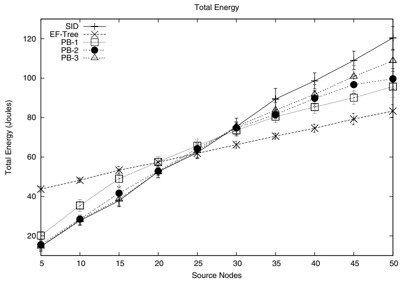


Fig. 7. Energy with routing rule 1

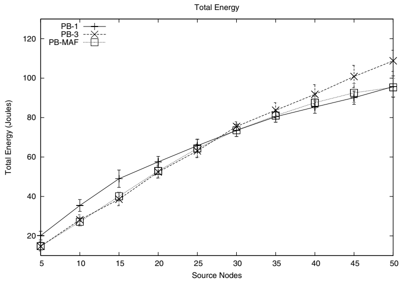


Fig. 8. Energy with routing rule 2

SID, because proactive behavior is rarely taken with the simulated parameters, thus their results were omitted.

The results for PB-2 and PB-3 are closer to the SID algorithm when the number of source nodes is less than 30. When a low traffic is measured, such policies do not assume the proactive behavior. Some differences are due to the random nature of the traffic, because source node data generations can be concentrated in a time interval without implying in a high occurrence ratio. In these cases, the performance of EF-Tree and PB-1 are not so good because they assume an unnecessary proactive behavior. But, when traffic increases, the EF-Tree and PB-1 have a better performance than PB-2 and PB-3 because their proactive behaviors avoid the cost of creating a routing infrastructure for new sources. PB-1 never reaches the EF-Tree performance for situations of high traffic since it always starts with a reactive mode, before changing to a proactive behavior.

In the previous results, the management entity perceives that the chosen policy is not so precise in the traffic evaluation. But, we can have advantages with the flexibility of the policy redefinition. For instance, that policy may be replaced by the one found in Fig. 5 that applies a Moving Average Filter (MAF). Fig. 8 shows the results of the new policy called PB-MAF. We can see that the PB-MAF results are closer to the best performance of PB-1 and PB-3, thus it is a more adaptable solution to respond to traffic changes by autonomously adjusting to its behavior.

5.2 Adaptive Routing Optimization

In this case study, we consider a continuous traffic scenario where the rule of Fig. 6 is applied to detect failures and rebuild the routing tree. We fixed the number of source nodes in 20, randomly chosen to send their data periodically towards the sink from the beginning to the end of the simulation. We varied the node failure ratio from 0 to 0.012 failures per second randomly distributed during the simulation. When a node fails, it stays inactive until the end of simulation. Fig. 9 shows the delivery rate improvement of the adaptive rule solution (EF-Tree adap) compared with the EF-Tree with fixed rebuildings (EF-Tree orig). It shows the advantage of failure detection to take an adaptive behavior. Fig. 10 shows the relative energy usage of the solutions. We note the better performance

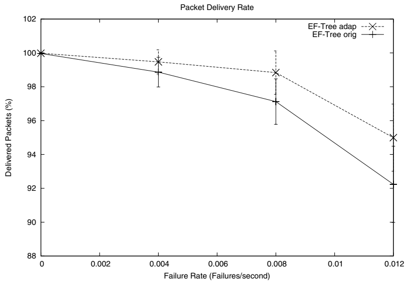


Fig. 9. Delivery rate with optimization rule

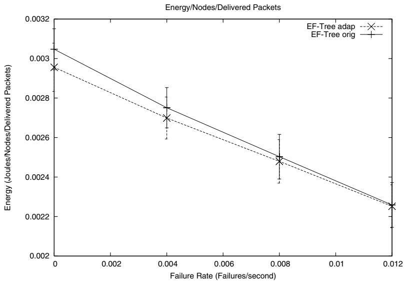


Fig. 10. Energy usage with optimization rule

of the adaptive solution when the failure rate is low due to rare tree rebuildings. However, with higher failure rates, the energy usage between the adaptive and original solutions approximates because more tree rebuidings are needed by the latter to maintain the delivery rate.

5.3 Policy Redefinition Cost

The advantage of redefining a policy depends on its gain versus the cost of its redefinition and redistribution. To take an idea of this cost, we assumed a byte-coded script of 20 bytes (large enough to store the script of Fig. 3(b)), totalizing an estimated packet size of 36 bytes with the control header. We used the classical flooding (broadcasts in the network) to redistribute policies to the entire network. The energy cost of a single distribution was 1.29 Joules, which is compensated by the difference of the PBs performance with different thresholds or with the dynamic one. In addition, this advantage may be higher if the network remains more time in the new situation.

5.4 Hierarchical Structure Evaluation

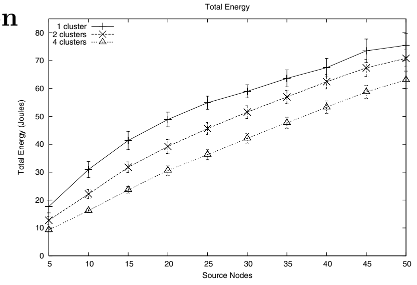


Fig. 11. Energy with 1, 2 and 4 clusters

To show the gain of a hierarchical scheme and the independent application of policies, we created a scenario like the event-driven where the sensing area is divided in quadrants defining separated clusters. We compare the performance of the PB scheme running with 2 and 4 clusters with a scenario with no clustering. Fig. 11 depicts the energy consumed by the algorithms with different number of source nodes. We observe a better performance as the number of clusters grows since independent actions are taken by their cluster-head's policies. An event may be restricted to a location and its detection in a partition does not imply in its detection in another area. Thus, a high traffic in a partition activates the proactive behavior only inside itself, whereas other partitions may keep a reactive behavior saving their resources.

6 Final Considerations

In WSNs, routing algorithms are specific for the scenario of a given application. In scenarios of high variability, a given routing algorithm cannot achieve its best performance all the time. Thus, adaptive hybrid approaches, providing autonomic behavior, can be better than a single algorithm. This work discussed the usage of policies to establish adaptive routing rules for WSN elements to become more flexible and accessible development and maintenance tasks of a network. Although we focused on low-level policies, they are necessary for high-level policy building. Case study scenarios revealed that the usage of policies in autonomous WSNs can provide resource savings.

The next step of our work is to enable and evaluate the implementation of policy-based routing rules on real sensor nodes. Future work includes the implementation of policy-based routing rules to change the routing QoS metric, and the use of policies to dictate how the data is collected, aggregated and forwarded through the routing infrastructure with resource savings.

References

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cyirci, *Wireless sensor networks: A survey*, Computer Networks, 38 (2002), pp. 393–422.
2. J. N. Al-Karaki and A. E. Kamal, *Routing techniques in wireless sensor networks: a survey*, IEEE Wireless Communications, 11 (2004), pp. 6–28.
3. A. Boulis, C.-C. Han, and M. B. Srivastava, *Design and implementation of a framework for efficient and programmable sensor networks*, in MobiSys, May 2003.
4. I. CISCO Systems, *White paper: Policy-based routing*, Access: May 2005. [Online] Available: http://www.cisco.com/warp/public/cc/techno/protocol/tech/.
5. D. Clark, *RFC 1102: Policy Routing in Internet Protocols.* MIT Lab for Computer Science, 1989.
6. Crossbow, *Mica2 platform*, Access: February 2004. [Online] Available: http://www.xbow.com/.
7. C. M. Figueiredo, E. F. Nakamura, and A. A. Loureiro, *Multi: A hybrid adaptive dissemination protocol for wireless sensor networks*, in Algosensors, vol. 3121 of Lecture Notes in Computer Science, Turku, Finland, July 2004, Springer, pp. 171–186.
8. S. Hedetniemi and A. Liestman, *A survey of gossiping and broadcasting in communication networks*, Networks, 18 (1988), pp. 319–349.
9. J. Heidemann, F. Silva, and D. Estrin, *Matching data dissemination algorithms to application requirements*, in 1st SenSys, Los Angeles, CA, USA, 2003, ACM Press, pp. 218–229.
10. W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, *Energy-efficient communication protocols for wireless microsensor networks*, in 33rd HICSS, Maui, Hawaii, USA, January 2000.

11. C. T. Inc., *Mote in-network programming user reference*, Access: August 2004. [Online] Available: http://webs.cs.berkeley.edu/tos/tinyos-1.x/doc/xnp.pdf.
12. J. O. Kephart and W. E. Walsh, *An artificial intelligence perspective on autonomic computing policies*, in 5th Int'l Workshop on Policies for Dist'd Systems and Networks, 2004.
13. M. Kochhal, L. Schwiebert, and S. Gupta, *Role-based hierarchical self organization for wireless ad hoc sensor networks*, in Proc. of the 2nd ACM Int'l Conf. on Wireless Sensor Networks and Applications, ACM Press, 2003, pp. 98–107.
14. R. Krishnan and D. Starobinski, *Message-efficient self-organization of wireless sensor networks*, in IEEE WCNC 2003, March 2003, pp. 1603–1608.
15. P. Levis and D. Culler, *Maté: A tiny virtual machine for sensor networks*, in 10th Int'l Conf. on Architectural Support for Prog. Lang. and Operating Sys., ACM Press, 2002, pp. 85–95.
16. P. Levis, N. Patel, D. E. Culler, and S. Shenker, *Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks.*, in 1st NSDI, 2004, pp. 15–28.
17. E. F. Nakamura, C. M. Figueiredo, and A. A. Loureiro, *Information fusion for data dissemination in self-organizing wireless sensor networks*, in 4th ICN, April 2005.
18. NS-2, *The network simulator - ns-2*, Access: February 2004. [Online] Available: http://www.isi.edu/nsnam/ns/.
19. C. Perkins, E. Belding-Royer, and S. Das, *Ad-hoc on-demand distance vector routing.* RFC 3561, 2003.
20. C. Perkins and P. Bhagwat, *Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers*, in ACM SIGCOMM'94, 1994, pp. 234–244.
21. G. J. Pottie and W. J. Kaiser, *Wireless integrated network sensors*, Communications of the ACM, 43 (2000), pp. 51–58.
22. V. Ramasubramanian, Z. Haas, and E. Sirer, *SHARP: A hybrid adaptive routing protocol for mobile ad hoc networks*, in 4th MobiHoc, 2003, pp. 303–314.
23. C. Shen, C. Srisathapornphat, and C. Jaikaeo, *Sensor information networking architecture and applications*, IEEE Personal Communication, 8 (2001), pp. 52–59.
24. M. Sloman, *Policy driven management for distributed systems*, Journal of Network and Systems Management, 2 (1994), pp. 333–360.
25. S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, San Diego, CA, USA, 2nd ed., 1999.
26. K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, *Protocols for self-organization of a wireless sensor network*, IEEE Personal Communications, 7 (2000), pp. 16–27.
27. S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, *A taxonomy of wireless micro-sensor network models*, ACM Mobile Computing and Communications Review (MC2R), 6 (2002), pp. 28–36.
28. A. Woo, T. Tong, and D. Culler, *Taming the underlying challenges of reliable multihop routing in sensor networks*, in 1st SenSys, ACM Press, 2003, pp. 14–27.
29. C. Zhou and B. Krishnamachari, *Localized topology generation mechanisms for self-configuring sensor networks*, in IEEE Globecom, San Francisco, USA, December 2003.

Decentralized Computation of Threshold Crossing Alerts

Fetahi Wuhib[1], Mads Dam[1], Rolf Stadler[1], and Alexander Clemm[2]

[1] KTH Royal Institute of Technology,
Stockholm, Sweden
{fzwuhib, mfd, stadler}@kth.se
[2] Cisco Systems
San Jose, California, USA
alex@cisco.com

Abstract. Threshold crossing alerts (TCAs) indicate to a management system that a management variable, associated with the state, performance or health of the network, has crossed a certain threshold. The timely detection of TCAs is essential to proactive management. This paper focuses on detecting TCAs for network-level variables, which are computed from device-level variables using aggregation functions, such as SUM, MAX, or AVERAGE. It introduces TCA-GAP, a novel protocol for producing network-wide TCAs in a scalable and robust manner. The protocol maintains a spanning tree and uses local thresholds, which adapt to changes in network state and topology, by allowing nodes to trade unused "threshold space". Scalability is achieved through computing the thresholds locally and through distributing the aggregation process across all nodes. Fault-tolerance is achieved by a mechanism that reconstructs the spanning tree after node addition, removal or failure. Simulation results on an ISP topology show that the protocol successfully concentrates traffic overhead to periods where the aggregate is close to the given threshold.

1 Introduction

Threshold crossing alerts (TCAs) indicate to a management system that some monitored MIB object, or management variable, has crossed a certain preconfigured value - the threshold. Objects that are monitored for TCAs typically contain performance-related data, such as link utilization or packet drop rates. In order to avoid repeated TCAs in case the monitored variable oscillates, a threshold is typically accompanied by a second threshold called the hysteresis threshold, set to a lower value than the threshold itself. The hysteresis threshold must be crossed, in order to clear the TCA and allow a new TCA to be triggered when the threshold is crossed again (Fig. 1).

TCAs represent an important mechanism in proactive management, as they allow for management that is event-based and does not need to rely as much on centralized polling.

Today, TCAs are generally set up per device, e.g., for monitoring packet drop rates on a particular link. In addition, Service Level Agreements (SLAs) are often articulated similarly, on a per-device basis, reflecting the limitations of today's technology. However, there is a definitive need for management functionality that provides cross-device TCAs, which are applied to parameters that are aggregated across the network. Examples include management applications that alert an operator whenever (a) the average link utilization in a domain rises above certain threshold, or (b) whenever the number

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 220–232, 2005.

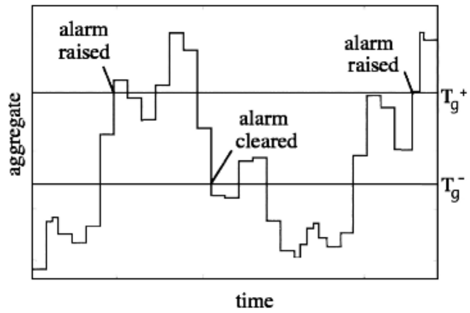


Fig. 1. Threshold Crossing Alerts: an alert is raised when a network-wide variable, the aggregate, exceeds a given global threshold T_g^+. The alert is cleared when the aggregate has decreased below a lower threshold T_g^-.

of currently active voice calls on a network, as aggregated across IP PBXs or voice gateways, exceeds a given value.

This work focuses on supporting TCAs for thresholds on network-wide management variables, which are computed by aggregating local variables across many devices. We will refer to such TCAs as *network TCAs (NTCAs)* and to network-wide management variables as *aggregates*. Typical NTCAs involve aggregates that are computed from device variables, using functions, such as SUM, AVERAGE, COUNT, MAX, or MIN. (For a discussion on the practical relevance of NTCAs, see [1].)

The hard part in determining when to trigger NTCAs is to ensure scalability and fault-tolerance of the approach. Traditionally, the aggregation of local variables from different devices has been performed in a centralized way, whereby an application, running on a management station, first retrieves state variables from agents in network devices and then aggregates them on the management station. Such an approach has well-known drawbacks with respect to scalability and fault tolerance.

We propose that NTCAs be computed in a distributed way. To this end, we assume that each network device participates in the computation by running a management process, either internally or on an external, associated device. These management processes communicate via an overlay network for the purpose of monitoring the network threshold. Throughout the paper, we refer to this overlay as the *network graph*. A node in this graph represents a management process together with its associated network device(s). While the topology of this overlay can be chosen independently from the topology of the underlying physical network, we assume in this paper, for simplicity, that both topologies are the same, i.e., that the management overlay has the same topology as the as the underlying physical network.

A straightforward solution to the NTCA problem can be constructed by using a protocol for distributed state aggregation, such as [2,3]. These protocols provide a continuous estimate of the network-wide aggregate on a dedicated root node, by setting up a spanning tree on the network graph, along which updates are reported. NTCAs can be detected by a filter on the root node. However, such a solution is inherently inefficient in terms of traffic overhead on the network graph and processing load on the management nodes. For the purpose of triggering NTCAs, we are not interested in receiving estimates about the dynamically changing aggregate, but only in receiving alarms when it crosses certain thresholds. For instance, no estimate of the aggregate is needed if its value is well below a threshold.

In this paper, we present TCA-GAP, a novel protocol for computing NTCAs in a scalable and robust manner. The protocol is based on the Generic Aggregation Protocol (GAP), which provides support for creating and maintaining a spanning tree on the network graph and for distributed aggregation of local variables [3]. The basic idea behind our protocol is the use of local thresholds that control whether a node reports a change in aggregate of its subtree. These thresholds are locally recomputed whenever local threshold rules are violated, which can be triggered, e.g., by a "significant change" in a device variable or a node failure. Scalability is achieved through computing the thresholds locally and through distributing the aggregation process across all nodes of the spanning tree. Fault-tolerance is achieved by a mechanism that reconstructs the spanning tree after node addition, removal or failure. We evaluate the protocol on an ISP topology and compare its performance to a naïve solution to the NTCA problem and to a centralized scheme for NTCA detection proposed by Dilman and Raz [4].

The paper is organized as follows. Section 2 reviews related work. Section 3 formally defines the NTCA problem. Section 4 provides GAP in a nutshell, and section 5 presents our protocol. Section 6 gives simulation scenarios, simulation results and a discussion of those results. Finally, section 7 provides some additional comments to the results and gives an outlook on further work.

2 Related Work

Dilman and Raz [4] study the NTCA problem for a centralized management system, where the management station communicates directly with the network elements. The authors assume that the aggregation function is sum and that a single global threshold T is given. In one solution, which the authors call 'simple-value', local threshold value of T/n where n is the number of nodes in the network is assigned to all nodes. Whenever the local weight becomes larger than this threshold, the node sends a trap with the current weight to the management station. Periodically, if the management station has received traps during the previous period, it polls all nodes that did not send a trap for their local weights. Then, it aggregates the local weights of all nodes and determines whether the global threshold has been crossed. This scheme performs well for "small" networks, where polling is feasible, where weights are evenly distributed, and where the likelihood of a node exceeding its threshold is small. In 6 we compare the performance of TCA-GAP to this simple-value scheme for a specific scenario. In the same paper, the authors propose a second scheme, called 'simple-rate', which assumes an upper bound on $\Delta w/\Delta t$, the range of change of weights per unit time. This assumption leads to an upper bound on the aggregate and thus allows nodes to be sampled less frequently.

Decentralized solutions for problems closely related to the NTCA problem have been proposed by Breitgand, Dolev and Raz in the context of estimating the size of a multicast group [5]. There, the task is to determine whether the group size is within a prescribed interval for which pricing is constant. Several schemes are proposed, based on the concept that nodes intercept traps generated by their children, in order to suppress false alerts.

Outside the specific domain of TCA generation, the problem of distributed state aggregation has recently received considerable attention (cf. [6, 7, 2, 8, 9]). An approach common to several authors (ourselves including) is to reduce traffic load by installing filters at each node of the aggregation tree. Olston et al. [10] propose a scheme whereby filters installed at each node continually shrink, leaving room for a central processor to reallocate filter space where needed most. This scheme was later refined in [11], by using statistics on the local aggregates held at each node, in order to allow filters to dynamically adjust to the data sources. One drawback of this approach is that it is not *temporally local*, because the criteria for setting the filters depend on the histories of previously sampled values. This makes the approach vulnerable to failures and dynamic changes, since these can affect the shape of the aggregation tree in unpredictable ways. (This approach though is *spatially local*, since each node makes local decisions to set the filters for its children.) By way of comparison, the solution we propose is both spatially and temporally local and applies a rather simple, history-free scheme to transfer threshold space between siblings in an aggregation tree.

3 The NTCA Problem

We are considering a dynamically changing network graph $G(t) = (V(t), E(t))$ in which nodes $i \in V(t)$ and edges/links $e \in E(t) \subseteq V(t) \times V(t)$ may appear and disappear over time. To each node i is associated a *weight*, $w_i(t) \geq 0$. The term weight is used to represent a local state variable or a device counter that is being subjected to threshold monitoring. For the main part of the paper we assume that weights are integer valued quantities, aggregated using SUM. In section 7 we discuss extensions to support other aggregates such as those mentioned in section 1.

The objective is to raise an alert on a distinguished root node, the management station, when the aggregate weight $\Sigma_i w_i(t)$ exceeds a given global threshold T_g^+, and to clear the alert on the root when the aggregate has decreased below a lower threshold T_g^-.

The design goals for the protocol are as follows:

- *Scalability:* the protocol must scale to networks of very large size. To this end, the protocol must ensure that the load on nodes and links is small and evenly distributed. In addition, for practical network topologies, the maximum processing load on each node and the maximum traffic load on each link should increase sublinearly with the network size.
- *Accuracy:*the accuracy requirement is subdivided into the following
  - (Soundness) An NTCA is raised only if the aggregate crosses T_g^+, and cleared only if the aggregate falls below T_g^-;
  - (Accuracy) An NTCA is raised if the upper threshold T_g^+ is exceeded for a certain minimal duration Δt_{alert}. For clearing the NTCA, the condition is symmetric;
  - (Timeliness) If an NTCA is raised (cleared), then it is raised (cleared) within some given minimal time t_{delay} after the relevant threshold crossing.
- *Robustness:* the protocol must adapt gracefully to changes in the underlying network topology, including node and link failures. This means that, for practically relevant scenarios involving node failures and/or topology changes, the protocol must produce output that is of practical use.

The Δt_{alert} condition is needed in order to adequately disregard transient behavior. A strict solution that does not allow for some such form of temporal imprecision cannot in fact be realized (cf. [12]). The soundness and timeliness conditions need to be interpreted in a similar way. We turn to this issue in section 7.

4 The GAP Protocol

The TCA-GAP protocol introduced in this paper is based on an earlier protocol, GAP - Generic Aggregation Protocol [3], for building and maintaining aggregation trees. GAP is a modified and extended version of the BFS (Breadth First Spanning) tree construction algorithm of Dolev, Israeli, and Moran [13]. The protocol of [13] executes in coarsely synchronized rounds, where each node exchanges with its neighbors its belief about the minimum distance to the root and then updates its belief accordingly. Each node also maintains a pointer to its parent, through which the BFS tree is represented.

The above work by Dolev et al. [13] exhibits similarities to the 802.1d Spanning Tree Protocol (STP) [14, 15]. STP is a distributed protocol that constructs and maintains a spanning tree among bridges/switches, in order to interconnect LAN segments. Similar to [13], a node in STP chooses its parent, such that its distance (measured in aggregate link costs) to the root node is minimized. The initialization phase though is very different between the two protocols. While STP uses broadcast in LAN segments and a leader election algorithm to determine the root node, [13] assumes a given root node and an underlying neighbor discovery service. Also the failure discovery mechanism is very different in both protocols.

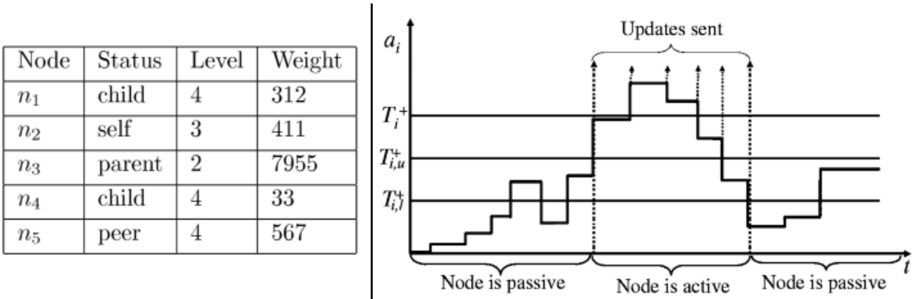
GAP extends [13] in a number of ways. First, GAP uses message passing instead of shared registers. Second, in GAP, each node maintains information about its children in the BFS tree, in order to compute the partial aggregate, i.e., the aggregate value of the weights from all nodes of the subtree where this node is the root. Third, GAP is event-driven. That is, messages are only exchanged as results of events, such as the detection of a new neighbor, the failure of a neighbor, an aggregate update, a change in local weight or a parent change. Fourth, since a purely event-driven protocol can cause a high load on the root node and on nodes close to the root, GAP uses a simple rate limitation scheme, which imposes an upper bound on message rates on each link.

In GAP, each node maintains a neighborhood table shown in Fig. 2(a), associating a status, a level, and an (aggregate) weight to each neighboring node. The status field (with values self, child, parent and peer) defines the structure of the aggregation tree. The value peer denotes a neighbor in the network graph that is not a neighbor in the aggregation tree. The level field indicates the distance, in number of hops, to the root. It is used to construct the BFS aggregation tree, whereby each node chooses its parent in such a way that its level is minimal. The weight field refers to the cached partial aggregate for a neighboring node and to the local weight for the local node.

GAP relies on underlying failure and neighbor discovery services, which are assumed to be reliable.

5 TCA-GAP: A Distributed Solution to the NTCA Problem

TCA-GAP assumes a designated root node to report NTCAs. Upon starting the protocol, the root node will map the global thresholds into local thresholds for its children on the aggregation tree, and each child will then, recursively, assign local thresholds to its own children. During the operation of the protocol, each node that has an aggregate



Node	Status	Level	Weight
n_1	child	4	312
n_2	self	3	411
n_3	parent	2	7955
n_4	child	4	33
n_5	peer	4	567

(a) Sample neighborhood table for GAP

(b) Local hysteresis mechanism in TCA-GAP that decides whether a node is active or passive.

Fig. 2. The GAP neighborhood table and the local hysteresis mechanism in TCA-GAP

far below (or above) the local threshold will enter a passive state, where it refrains from forwarding changes of its aggregate to its parent. Once a node's aggregate gets close to its local threshold, it will become active and start reporting changes of the aggregate to its parent. A passive node will adapt to changes in network state and to failures, by dynamically recomputing the local thresholds of its children. In the following, we describe the main features of TCA-GAP in more detail.

Local Hysteresis Mechanism: A local hysteresis mechanism determines whether a node sends updates of its aggregate to its parent. A node that sends updates is called *active*. One that does not send updates is called *passive*. The local hysteresis mechanism is similar to the global hysteresis mechanism (see Fig. 1), but it serves a different purpose, namely, to correctly sample the aggregate when the global threshold is crossed. The local threshold assigned to node i is T_i^+. From T_i^+ the upper and lower local hysteresis thresholds $T_{i,u}^+$ and $T_{i,l}^+$ are computed, such that $T_{i,u}^+ = k_1 T_i^+$ and $T_{i,l}^+ = k_2 T_i^+$. Here, k_1 and k_2 are some *global control parameters*. Using these threshold values, a node will decide when to send its aggregate value as shown in Fig. 2(b). The transition between active and passive states occurs as follows. When the local aggregate of a passive node grows beyond the upper threshold $T_{i,u}^+$, the node becomes active. It will stop performing threshold recomputations (see below), start sending aggregate updates (just as in the GAP protocol), and reset the thresholds of its children to 0. When the aggregate of an active node decreases below $T_{i,l}^+$, then the node becomes passive and recomputes the thresholds of its children. The threshold of child j with aggregate a_j will be set to $T_i^+ * (a_j/a_i)$, where a_i is the aggregate of the local node.

Threshold Rules: The (local) threshold rules guarantee that, if a global threshold is crossed, then there is at least one node, for which these rules are violated. They are:

(1) $T_i^+ \geq \Sigma_{j \in J} T_j^+$ where J is the set of children of node i;

(2) If J' is the set of active children, then $\Sigma_{j \in J'} T_j^+ \geq \Sigma_{j \in J'} a_j$ where a_j is the local aggregate reported by child j.

As long as these two rules remain valid on a node, it stays passive. Once one of them is violated, the node will recompute the thresholds of its children.

Threshold Recomputation: threshold recomputation allows an active node to "receive threshold space" from one or more passive siblings. The purpose of this is to reinstate the threshold rules on the node.

It is generally difficult to recompute thresholds with a small overhead. For instance, a greedy approach to threshold recomputation will attempt to give an active child enough threshold space to make it passive. Such a solution, however, is prone to oscillation, since it can lead to two children alternately borrowing threshold space from each other. For this reason, we take a conservative approach that allows an active child of a passive node, under certain conditions, to remain active, without threshold recomputation having to occur.

Recomputation of local thresholds can happen for two reasons:

- Event #1: a node receives from its parent a new, lower threshold T', causing threshold rule (1) above to fail;
- Event #2: a child reports an increased aggregate, causing threshold rule (2) to fail.

These events can have several causes. For instance, the change of a local weight in some subtree can cause event #2 at the root of that subtree. When a device fails or is removed from the network, the topology of the aggregation tree may change, which in turn may cause events #1 or #2 to occur at different nodes in the network. The same can happen in the case where a device recovers from failure or is added to the network.

As a reaction to one of the above events, a node recomputes the local thresholds as follows:

i. For event #1, we reduce the threshold of one or more passive children by $\Sigma_{j \in J} T_j^+ - T'$, where J is the set of children of the current node. Observe that, if such a reduction is not possible, then T' will be less than the sum of the thresholds of the active children, and, therefore, has reverted to active state.
ii. For event #2, we reduce the threshold of one or more passive children by some value $\delta > \Sigma_{j \in J'} a_j - \Sigma_{j \in J'} T_j^+$, where J' is the set of active nodes, and increase the assigned threshold value of an active child by the same amount.

In case [ii.], there is some freedom in choosing δ and the child j whose threshold we increment. However, δ should not exceed $\frac{a_j}{k_2} - T_j^+$, since, as we noted, there is risk for oscillation. For our protocol, we choose j such that $a_j - T_j^+$ is maximized, and we choose $\delta = \frac{a_j}{k_1} - T_j^+$. Observe that, since the threshold rules are evaluated at the end of each protocol cycle, only an aggregate update from a single child can have caused event #2. Therefore, some j can be found, such that the resulting δ will satisfy [ii.]. To reduce the threshold of the passive children by δ, the threshold of each passive child is reduced, in turn, in the order of decreasing thresholds. This solution attempts to minimize the threshold updating overhead at the cost of a substantial risk that at least one child will become active in the next round.

Topology Changes and Failures: When a node is removed or fails, the protocol follows the GAP design, i.e., the failure detector informs all neighbors, and, as a result, the parent updates its neighborhood table by removing the failed node and recomputing its aggregate accordingly. When a new node is discovered, or a failed node recovers, the protocol again follows the GAP design by attaching it to a suitable parent. The parent receives an update message from the new node, creates an entry for the node in the neighborhood table, and updates the aggregate accordingly. In all of the above cases, the threshold rules are evaluated at the end of the protocol cycle, which may include threshold recomputation, etc., as described above.

Symmetric Modes: Once the aggregate exceeds T_g^+, then all nodes will have become active, and the overhead of TCA-GAP increases to that of GAP. To a large extent, this problem of a large overhead can be addressed by exploiting the inherent symmetry in the NTCA problem: detecting an upwards crossing of an upper threshold level is not different from detecting a downwards crossing of the lower one. In the first case, nodes will be passive on small aggregates and set to trigger when aggregates become large. In the second case, nodes will be passive on large aggregates and trigger when aggregates become small. Reflecting this, the protocol can work in one of two symmetric *modes*, positive or negative, depending on which threshold and which direction of threshold crossing it is set to trigger. The switch between modes is done at the root, as a result of global threshold crossings, and propagated down the aggregation tree, by adding the mode to the update messages exchanged between neighbors. Locally, each node i is assigned either an upper threshold T_i^+ or a lower threshold T_i^-. In positive mode, the objective is to detect upwards crossings of T_i^+. In negative mode, the objective is symmetric, i.e., to detect downwards crossings of T_i^-. The local hysteresis thresholds in the latter case are computed as $T_{i,u}^- = T_i^-/k_1$ and $T_{i,l}^- = T_i^-/k_2$. Note that, for simplicity of presentation, the discussion in the previous subsections refers to the positive mode. The extension to negative mode is straightforward.

Initialization: The protocol initializes in the same way as GAP, which constructs the BFS spanning tree and populates the local neighborhood table [3]. As part of this initialization process, the local thresholds in all nodes are set to 0 in positive mode, causing weight changes to be reported up the aggregation tree to the root node. In the second phase of the initialization process, the root node sets its two global thresholds as instructed by the management station, which causes the recomputation of local thresholds to be propagated from the root down the aggregation tree. As a result, nodes start filtering weight and aggregate changes.

Code: The main data structure manipulated by TCA-GAP is the neighborhood table, which, in addition to the four columns of Fig. 2(a), contains a fifth column for thresholds. All numerical fields in the neighborhood table are arbitrarily initialized to 0. The update vector in TCA-GAP serves a similar purpose as in GAP, namely, informing a neighbor of a node about changes in the node's neighborhood table. This vector is a tuple of the form `(update,From,Weight,Level,Parent,ThresholdList, Sign)` where `ThresholdList` is a list of (node, threshold) pairs, and `Sign` is the mode. The protocol assumes an external, virtual root with level 0. Further, it assumes underlying services for failure detection and neighbor discovery. Local weight changes are handled by assuming that two instances of TCA-GAP run on every node, one, a leaf, for local weight changes, and one for aggregation. The main operation embodying the TCA-GAP semantics is the function `restoreTableInvariant`, which is responsible for maintaining the TCA-GAP invariants. These invariants ensure, for instance, that each node has a unique parent, and that the local threshold rules hold. In case the invariants are violated, actions are performed, such as selecting a new parent, switching between passive and active operation, or recomputing thresholds. The pseudo code for TCA-GAP can be found in [1].

6 Experimental Evaluation

We evaluated the functionality and performance of TCA-GAP through simulation, under varying topologies, thresholds, weight change and node failure models. The main hypotheses we wanted to test were:

- At low aggregate/threshold ratios the TCA-GAP scheme produces a management traffic overhead several orders of magnitude below that of a scheme for continuous monitoring such as GAP;
- The performance of TCA-GAP degrades gracefully as the aggregate/threshold ratio approaches 1.

The simulation results we have obtained are very encouraging. In the paper, we show results for two scenarios using an ISP network as the underlying topology. The first scenario involves several sinusoidal threshold crossings and shows how TCA-GAP successfully manages to reduce traffic when the aggregate is far from the thresholds. The second scenario shows the behavior of TCA-GAP at a low static aggregation level.

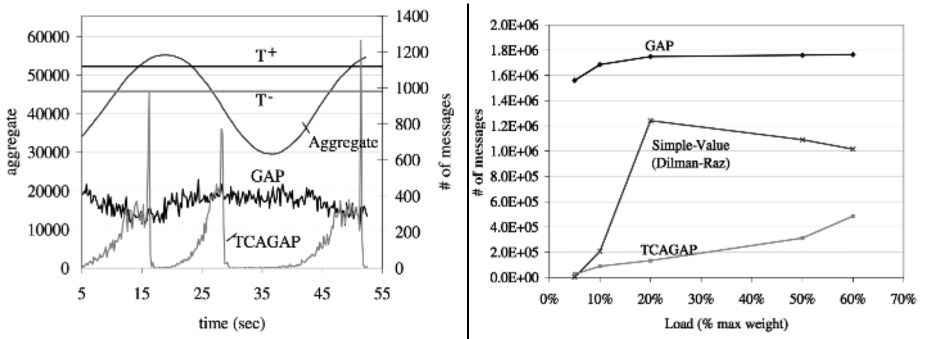
For the simulations we used a 221 node grid network topology and the topology of an ISP, Abovenet, from [16]. In this paper we mainly report on simulation results from Abovenet, a network with 654 nodes and 1332 links. The simulation studies were conducted with the SIMPSON network simulator [17]. The experiments were run with a uniform message size of 1Kbyte, a bandwidth of 100MB/sec, a processing delay of 1ms, and a communication delay of 4ms.

In the simulation runs, rate limitation was ignored to better compare the key properties of TCA-GAP versus the other two schemes. The main effect of rate limitation is to smooth peak traffic volumes, by imposing an upper bound on the traffic on each link. Secondly, the overall traffic is reduced since each node has the ability to process several messages before an output is produced.

Local weights are constrained to the interval $[0, \ldots, 100]$. Weight changes are simulated using a random walk model with random step sizes. Changes occur at randomly selected nodes, following a Poisson distribution with an average change rate per node of 1 weight change per second.

For the simulations shown in this paper, the global thresholds have been chosen at $T_g^+ = 80\%$ and T_g^- = 70% of the maximally achievable aggregate, i.e., the aggregate value where all nodes have the maximum possible weight of 100.

In the first scenario, nodes are initialized with weight values that are uniformly distributed in $[0, \ldots, 100]$ and the step size of the random walk model is biased with a sinusoidal input, such that $w_i(t + \Delta t) = w_i(t) + \Delta w + b * sin(t/\omega) + k$ where Δw is chosen to be uniformly distributed in an interval $[-x, \ldots, x]$, and the constants b and ω are chosen to obtain a suitable period and amplitude for the superimposed sinusoid on the aggregate. The constant k is a bias added to tune the aggregate to a desired long term average value. In Fig. 3(a), we show the aggregate, upper and lower global thresholds and, for both GAP and TCA-GAP, the total number of messages over a 200ms sampling period. The local thresholds are computed using $k_1 = 0.9$ and $k_2 = 0.85$. Three threshold crossings are shown in the figure. Each of them involves a gradually increasing number of active nodes, until a point is reached where all nodes are active. In the plot shown, this state is retained for a couple of seconds, until, after mode switching, the root reverts to passive state by crossing the relevant lower local threshold. (Note that, for any given node i in negative mode, the 'lower' threshold $T^{-}_{i,l}$ is actually higher than the 'higher' threshold $T^{-}_{i,u}$.) During the interval where all nodes are active, TCA-GAP behaves roughly as GAP. The large peaks are due to the root node propagating new threshold values, along with the sign, down the aggregation tree.



(a) Scenario 1: Messages generated by TCA-GAP and GAP; global aggregate over time; upper and lower values of global thresholds

(b) Total number of messages produced during a simulation run for TCA-GAP, SV and GAP in function of the average global aggregate

Fig. 3. Simulation Results

An interesting feature of TCA-GAP which is brought out in Fig. 3(a) is that traffic is concentrated around the global threshold crossings, and *not* where aggregates are maximal. This is attractive, since large aggregates are often indicative of congestion or overload situations in the network where it is desirable to keep management overhead to a minimum.

In the second scenario, the average aggregate weight has been chosen to be well below the global threshold, at about 5%. For modeling the weight changes, we used a biased version of the random walk model, such that $w_i(t + \Delta t) = w_i(t) + \Delta w + k$, where Δw is chosen to be uniformly distributed in an interval $[-x, \ldots, x]$.

The simulation results show that, on average, TCA-GAP generates 1.7% of the messages that are generated by GAP. At any given time, 95% of the nodes were passive and therefore did not produce messages. Graphs of the simulation results can be found in [1].

For the second scenario, we performed similar simulation studies with higher variability of the weight changes. As expected, the traffic and processing overhead of TCA-GAP was larger than above, but still substantially smaller than that of GAP.

For both of the above scenarios, we performed the same simulations for a 221 node grid network. The simulation results were similar to those for the Abovenet topology, used in scenarios 1 and 2.

We have also compared our scheme to other schemes for detecting threshold crossing alerts [4] in the number of messages consumed in the whole network. To compare the performance of TCA-GAP with the simple-value scheme (SV) by Dilman-Raz [4], we performed a number of simulations using scenario 2 with various average aggregates. Fig. 3(b) shows the results of the simulations. It gives the number of messages generated by TCA-GAP, SV and GAP, as a function of the average aggregate during the simulation run of 900secs. As one can see, in GAP, the number of messages is hardly affected by the level of the aggregate. We explain this by the fact that, this protocol does not attempt any filtering and propagates all changes, no matter how small. The SV scheme by Dilman and Raz distributes static thresholds to all nodes, and determines whether a global TCA has occurred by polling all nodes whenever a node reports crossing of its locally assigned threshold level. From the measurements, we conclude that, for aggregate levels of less than 5%, SV produces a low number of messages compared to both GAP and TCA-GAP, since local thresholds are almost never crossed. However, as the aggregate level grow larger than 10%, the probability of local threshold crossings increases significantly, which is reflected by a large increase in traffic volume. The subsequent reduction in traffic for SV is due to the fact that, in SV, nodes sending traps do not subsequently need to be polled. For TCA-GAP the traffic volume increases at a much smaller rate.

7 Discussion and Future Work

In this section, we evaluate our simulation results against the design goals of TCA–GAP concerning accuracy, scalability and robustness set out in section 3, and point to areas of future work.

Accuracy: As we have pointed out, a protocol that does not allow some temporal imprecision cannot be engineered. In particular, in a practical system, threshold crossings must have some minimum duration Δt_{alert} to guarantee detectability. Moreover, for a deterministic solution such as TCA-GAP, threshold detection can only be guaranteed when the network is stable: it is not hard to come up with failure patterns that conspire to continuously reconfigure the aggregation tree, such that threshold crossings never get detected at the root. For a stable network an upper bound on Δt_{alert}, the amount of time a threshold must remain crossed for TCA-GAP to guarantee detection, is $\mathcal{O}(n * (t_d + t_l) * h)$ where:

- n is the degree of the network graph = maximal number of neighbors for any node;
- t_d is the rate limitation timeout;
- t_l is the maximum communication delay between nodes;
- h is the height of the aggregation tree = the network diameter.

This bound is obtained, since, in the worst case, the threshold crossing has to be propagated to the root from the furthest distant leaf. Along each node, threshold violation has to be propagated through all neighbors in turn. Observe that the parameters n and h are determined by the choice of the management overlay topology, which, for this paper, is chosen to be identical to the underlying physical network topology. *Soundness*, in that NTCA's are raised only if the aggregate actually crosses T_g^+, is conjectured to hold in similar terms: if an NTCA is raised for at least the duration Δt_{alert} then it can be guaranteed that the threshold was also crossed some time during this interval. The *timeliness* requirement needs to be similarly adapted.

Scalability and Robustness: TCA-GAP uses a simple rate limitation scheme to impose an upper bound on the management traffic on each link. This by itself is not sufficient to guarantee scalability, however, unless a bound on the degree of nodes is also imposed. The effect of this is not trivial, and left for future work. For robustness, the protocol is certainly capable of adapting to topology changes in a graceful way. We confirmed this in simulations using scenarios with node failures and recoveries with results consistent with the other results we have reported.

Aggregation Functions: Above,we have used SUM as the aggregation function. Other simple aggregates like COUNT, MIN, and MAX can be supported with no modifications other than replacement of the aggregation function. For instance, to count the number of nodes with some local attribute exceeding c the local weight function $w_i(t)$ will return 1, if the attribute value exceeds c and 0 otherwise, and the aggregation function will be SUM. One way of handling AVERAGE in our framework could be to extend the underlying tree management protocol to maintain also node counts. This can be done by adding a further "tree size" field to the neighborhood table. Moreover, the cost of maintaining node counts only amounts to extending the topology update messages which are already exchanged by a node count field.

Future Work: For the evaluation of our protocol in this paper, we have used a random walk model to capture the fluctuations of the device variables, i.e., the local weights. While random walk models have been used before to model the changes of device variables (e.g., changes in load on host interfaces [10]), we plan to further evaluate TCA-GAP using real traces. Further, we plan on analyzing the effect of using different overlay topologies on the performance of TCA-GAP and on making our protocol resilient with regard to root failures. Also, we plan to investigate proactive threshold recomputation schemes and more complex aggregation functions. Finally, an implementation of TCA-GAP on the Weaver platform [18] is under way in our laboratory at KTH.

Acknowledgments. This work has been supported by a grant from Cisco Systems and a personal grant from the Swedish Research Council.

References

1. F. Wuhib M. Dam R. Stadler and A. Clemm. Decentralized computation of threshold crossing alerts. Technical report, KTH Royal Institute of Technology, 2005.
2. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGggretation service for ad-hoc sensor networks. In *Proc. 5th Symposium on Operating Systems Design and Implementation*, pages 131–146, 2002.
3. M. Dam and R. Stadler. A generic protocol for network state aggregation. In *Proc. Radiovetenskap och Kommunikation (RVK)*, 2005.
4. M. Dilman and D. Raz. Efficient reactive monitoring. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(4), 2002.
5. David Breitgand, Danny Dolev, and Danny Raz. Accounting mechanism for membership size-dependent pricing of multicast traffic. In *Networked Group Communication*, pages 276–286, 2003.
6. R. van Renesse. The importance of aggregation. In *In (A. Schiper, A.A. Shvatsman, H. Weatherspoon, and B. Y. Zhao, eds.), Future Directions in Distributed Computing, Lecture Notes in Computer Science*, volume 2584, pages 87–92. Springer-Verlag, 2003.
7. I. Gupta, R. van Renesse, and K. Birman. Scalable fault-tolerant aggregation in large process groups. In *Proc. Conf. on Dependable Systems and Networks*, pages 433–442, 2001.

8. David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 482, Washington, DC, USA, 2003. IEEE Computer Society.
9. Mohamed A. Sharaf, Jonathan Beaver, Alexandros Labrinidis, and Panos K. Chrysanthis. Tina: a scheme for temporal coherency-aware in-network aggregation. In *MobiDe '03: Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access*, pages 69–76, New York, NY, USA, 2003. ACM Press.
10. Chris Olston, Jing Jiang, and Jennifer Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 563–574, New York, NY, USA, 2003. ACM Press.
11. N. Roussopoulos A. Deligiannakis, Y. Kotidis. Hierarchical in-network data aggregation with quality guarantees. In *Proc. 9th International Conference on Extending Database Technology (EDBT)*, March 2004.
12. M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. In *Manuscript*, 2003.
13. S. Dolev, A. Israeli, and S. Moran. Self-stabilization of dynamic systems assuming only read/write atomicity. *Distributed Computing*, 7:3–16, 1993.
14. IEEE. *ANSI/IEEE Std 802.1D, 1998 Edition*. IEEE, 1998.
15. R. Perlman. *Interconnections, Second Edition*. Addison Wesley Longman, 2000.
16. N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *Proc. ACM/SIGCOMM*, 2002.
17. K. S. Lim and R. Stadler. SIMPSON — a SIMple Pattern Simulator fOr Networks. http://www.comet.columbia.edu/adm/software.htm, 2005.
18. K. S. Lim and R. Stadler. Weaver — realizing a scalable management paradigm on commodity routers. In *Proc. 8th IFIP/IEEE Int. Symp. on Integrated Network Management (IM 2003)*, 2003.

Control Considerations for Scalable Event Processing

Wei Xu[1], Joseph L. Hellerstein[2], Bill Kramer[1], and David Patterson[1]

[1] Computer Science Dept., University of California, Berkeley, CA
{xuw, pattrsn}@cs.berkeley.edu, kramer@lbl.gov
[2] IBM T.J. Watson Research Center, Hawthorne, NY, USA
hellers@us.ibm.com

Abstract. The growth in the scale of systems and networks has created many challenges for their management, especially for event processing. Our premise is that scaling event processing requires parallelism. To this end, we observe that event processing can be divided into intra-event processing such as filtering and inter-event processing such as root cause analysis. Since intra-event processing is easily parallelized, we propose an architecture in which intra-event processing elements (IAPs) are replicated to scale to larger event input rates. We address two challenges in this architecture. First, the IAPs are subject to overloads that require effective flow control, a capability that was not present in the components we used to build IAPs. Second, we need to balance the loads on IAPs to avoid creating resource bottlenecks. These challenges are further complicated by the presence of disturbances such as CPU intensive administrative tasks that reduce event processing rates. We address these challenges using designs based on control theory, a technique for analyzing stability, accuracy, and settling times. We demonstrate the effectiveness of our approaches with testbed experiments that include a disturbance in the form of a CPU intensive application.

1 Introduction

The advent of the Internet, sensor networks, and peer-to-peer networks has greatly increased the scale of distributed systems, making it more difficult to process events to detect and diagnose problems. Scaling event processing requires an architecture that incorporates parallelism. Herein, we address control challenges in providing such parallelism, namely: (a) providing flow control within replicated elements to avoid overload conditions and (b) balancing load in the presence of variable processing demands and other disturbances. Our solution to both of these challenges employs control theory, a formal approach to designing feedback loops.

Event streams consist of many kinds of data. For example, there are notifications of requests for service such as requests to a DNS (Domain Name Service) for name resolution; performance statistics such as response times; and trouble tickets that describe actions taken. These data are input to event processing

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 233–244, 2005.

components that detect abnormal situations, anticipate future problems, and diagnose existing problems.

Our motivation for scaling event processing comes from a company that is key to the eCommerce ecosystem. At the heart of this business is a DNS root server that generates events at a rate of 11 million to 42 million per hour. Many off-the-shelf products provide event processing capabilities, such as HP OpenView[5], IBM Tivoli[6] or Microsoft Operations Manager[7]. However, all are severely challenged by such high event rates.

Much related work exists in the area of event processing. Yemini *et al.* consider how to associate problem causes with symptoms using a code book algorithm [15]. Hofmeyr *et al.* develop techniques that discriminate between normal and abnormal operations [10]. Pinpoint System deals with the localization of failures on a production eCommerce system based on decision trees that analyze event data [3] and further extended to detect anomalies and failed components by automated analysis of execution paths in J2EE(Java 2 Enterprise Edition) applications [4]. Vilalta *et al.* predict critical events in computer system such as high CPU utilization and router failures by applying temporal data mining and time series analysis [14]. Burns *et al.* describe how to construct processing rules from event data [1]. These results identify requirements for event analysis, such as the need to have events in time serial order and to estimate accurately statistics such as the distribution of event sources and response times.

Supporting large scale event processing requires a scalable infrastructure. Astrolabe [13] and PIER [11] provide scaling by collecting and analyzing data on the nodes where they are generated. However, this approach limits the scope of the events analyzed to a single node. The Siena system [2] provides a publish/subscribe event-notification service with considerations for efficiencies and scaling. However, since this is a general infrastructure, it does not exploit the characteristics of event processing such as the opportunity to do intra-event processing in parallel.

From the foregoing, it seems that there has been little focus on scaling event processing. Thus, we introduce an approach that provides scaling through parallelism by identifying two kinds of processing that take place in event processing. Inter-event processing, such as problem diagnosis, analyzes multiple events in combination. Intra-event processing, such as filtering events from specific sources, considers events in isolation. Intra-event processing is easily parallelized by replicating the elements used for intra-event processing. We refer to these as intra-event processing elements (IAPs). We have encountered two challenges in scaling intra-event processing. First, IAPs are subject to overloads that require effective flow control, a capability that is often missing in off-the-shelf components. Second we must balance the load placed on IAPs to avoid bottlenecks. These challenges are further complicated by the presence of disturbances such as CPU intensive administrative tasks (e.g., Java Virtual Machine (JVM) garbage collection) that reduce event processing rates.

The remainder of this paper is organized as follows. Section 2 presents our architecture for scalable event processing. Section 3 applies control theory to designing key elements of a scalable event processing system. Section 4 reports the results of experiments we conducted to assess scaling. Our conclusions are contained in Section 5.

2 Architecture

This section describes our architecture for scalable event processing.

Event processing operates on the attributes of events and the relationships between these attributes. For example, performance events may have attributes such as *IP address*, *memory utilization*, *swap utilization*, and *load average.* For example, an event processing system might employ rules (or other representations) such as the following:

- Rule 1: Discard performance events from the subnet 92.126.10/24.
- Rule 2: Send an alert if the largest load average exceeds 2 for the hosts on subnet 92.126.11/24.

Rule 1 might be used to filter events from a test machine. Rule 2 is useful if all machines on the subnet 92.126.11 are production servers and we want to determine if there is a resource bottleneck.

Rules 1 and 2 suggest that there are two kinds of event processing: (1) intra-event processing such as filtering events that are not of interest and (2) inter-event analysis such as detecting a resource bottleneck. By definition, intra-event analysis is done on events in isolation. Inter-event analysis establishes relationships between events and so typically processes events in time serial order.

In the sequel, we focus on intra-event processing because of the opportunity to scale event processing by distributing the work to multiple nodes. Examples include: sampling events to obtain a representative distribution of response times; cleansing data to eliminate ill-formed events and unnecessary attributes; and augmenting events to associate the host name and host type based on host IP address. Through we only look at one event at a time, the processing can be expensive. We see that many of these operations require access to other information sources, such as a table that relates IP address to host name and type.

These observations led us to the two tier architecture that is depicted in Figure 1. Incoming events arrive in (rough) time sequence order. The first tier provides scalable intra-event processing, such as projections to eliminate unwanted attributes and joins to include additional attributes. There are three types of elements in this tier: the load splitter, the **I**ntr**A**-event **P**processing elements (IAP), and the combiner. Scaling is provided by having multiple IAPs that operate in parallel, possibly with different processing speeds and other characteristics. The load splitter assigns events to an IAP, and the combiner consolidates the results in time sequence order. We must be sure that we can implement the load splitter in a very efficient way so that it is not a bottleneck. As we show in Section 3.2, because the load splitter does not look into the event, it is much faster than the IAPs.

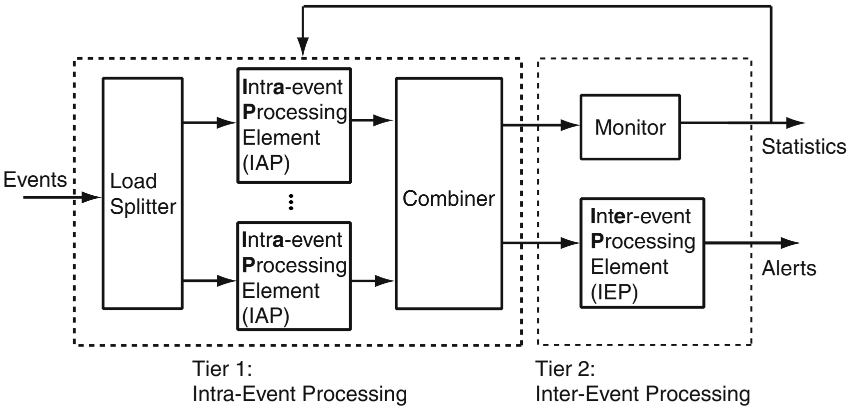

Events
Load Splitter
Intra-event Processing Element (IAP)
Intra-event Processing Element (IAP)
Combiner
Monitor
Statistics
Inter-event Processing Element (IEP)
Alerts
Tier 1:
Intra-Event Processing
Tier 2:
Inter-Event Processing

Fig. 1. A two tier architecture for scalable event processing. The first tier processes events in isolation. The second tier addresses relationships between events. Scaling is achieved in the first tier by having multiple IAP elements.

The second tier in Figure 1 performs inter-event processing. The **I**nt**E**r-event **P**rocessing element (IEP) inputs events in time serial order, and outputs alerts and higher level events. The monitor calculates statistics that are used as filtering criteria by the first tier, such as quantiles of response times contained in the attributes of incoming events that are used to identify exceptional situations.

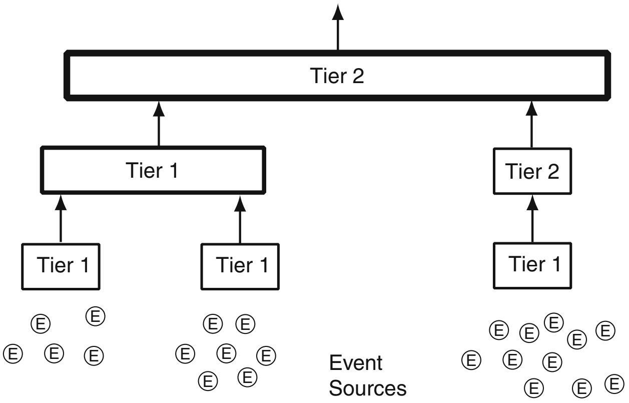


Fig. 2. Generalized architecture for scalable event processing

Figure 2 generalizes the architecture in Figure 1 to handle large scale distributed systems by treating tier 1 and tier 2 as components that can be replicated as needed. For example, network utilizations and communication delays are reduced by filtering events close to their origin. This argues for having instances of the first tier in many locations, such as satellite campuses and local area networks for critical servers. In contrast, a second tier instance may be quite distant from event sources in order to have a sufficient scope of events to do root cause analysis and obtain accurate statistical distributions. Thus, multiple first tier instances may feed into a single second tier. It may also be that there is a hierarchy of second tier instances, such as for event processing that occurs based on geographic scale (e.g., city, state, country). Thus, multiple second tier instances may input events to another second tier instance.

We focus on the requirements for scaling in the first tier. To better understand these issues, we implemented an IAP that embeds a TelegraphCQ (TCQ) system [12] to handle SQL based processing of event streams within the IAP. Our studies reveal two issues with increasing the event input rate. The first issue relates to flow control within IAP nodes. The second concerns balancing the loads of the IAPs.

3 Control Design

This section describes how we address issues in scaling intra-event processing, namely—(1) flow control within IAP nodes and (2) load balancing across IAPs.

3.1 Flow Control Within IAP Nodes

We begin by studying the effect of load on an IAP. Since our IAPs embed a TCQ, we represents events as data tuples in a object-relational schema and quantify throughput by using the TCQ metric tuples/sec, which is the same as events/sec. Figure 3 reports the results of experiments conducted on a single IAP node. We see that at moderate to heavy loads, tuples are dropped. This is problematic for two reasons. First, drops are not selected at random and so the presence of drops can bias the event statistics that are used for threshold-based filters and other purposes. Second, as we can see in the next paragraph, the drop happens after all processing on that tuple is done. Thus, dropping a tuple does not reduce workload on a server.

Going into more detail, a TCQ is structured into two parts: (a) a front-end process that interacts with requesters, parses inputs and translates them into internal data structures and (b) a set of back-end processes that perform relational database operations. The output of the back-end is placed into a *result queue* whose entries are retrieved by the front-end to respond to in-coming requests. The drops are a consequence of an overflow of the result queue, which is evident in Figure 3 since drops occur as the free space goes to 0.

One solution to the drops problem is to have front-end processes block when the result queue is full. Thus, the tuple is either dropped or throttled via admission control. Unfortunately, this can cause unpredictable effects on other queries because of the complex sharing that takes place. A second approach is to make the result queue very large to avoid having drops. But this means there is less memory available for front-end and back-end processes, which reduces throughput. A third technique is to do off-line experiments to determine the processing

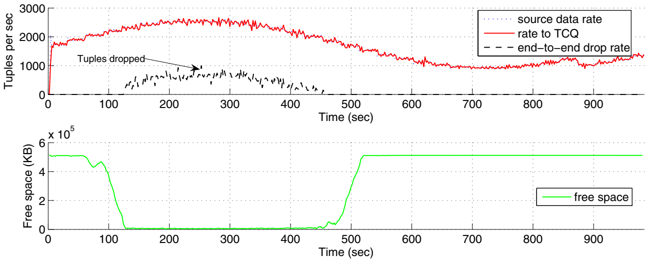


Fig. 3. Behavior of a TCQ node without regulating result queue length. The top plot shows the event input rate, and the drop rate. The bottom plot depicts the free space in the result queue. The drop rate increases with the event input rate.

capacity of an IAP node for a representative set of events. However, this is difficult to do because certain dynamics affect free space of the result queue, such as changes in the distribution of event types that in turn affects the amount of processing done (especially due to the selectivity of database queries).

Our approach to eliminating drops is to implement flow control within the IAPs by regulating the rate at which events are accepted by front-end processes. Thus, events are held in a queue within the IAP until there is sufficient space in the result queue. Such a design avoids the complexities of blocking front-end processes that hold resources associated with partially completed requests.

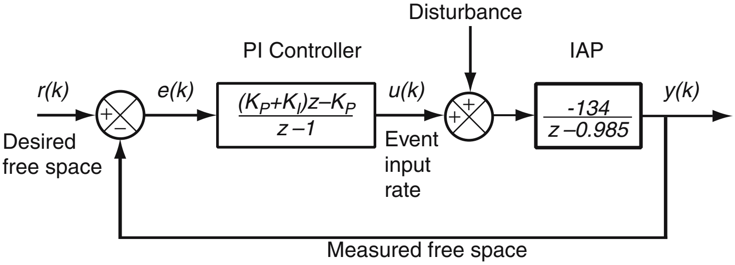


Fig. 4. Block diagram of IAP admission control

Figure 4 is a block diagram of the IAP flow control that we propose. This control system seeks to maximize throughput without dropping events by regulating the free space of the result queue. The reference input is the desired free space. The controller uses the difference between this reference and the measured free space of the result queue to adjust the event input rate.

We design the controller as described in [9]. The first step is to model how the event input rate affects free space of the result queue. Let $y(k)$ be the free space at time k, and let $u(k)$ be the event input rate. We use the first order model

$$y(k+1) = ay(k) + bu(k) \tag{1}$$

since [9] contains many examples in which this works well for real systems. Values of the parameters a and b are estimated from data obtained from studies of a testbed system, yielding $a = 0.985$ and $b = -134$. With this, we express the relationship in Equation (1) as a *transfer function*, a representation that expresses time serial effects in terms of z (which can be interpreted as a delay operator). The transfer function here is

$$\frac{-134}{z - 0.985}. \tag{2}$$

The transfer function in Equation (2) provides several insights. First, consider the transfer function's *steady state gain*, a quantity that indicates the effect of a small change in event input rate on free space. Steady state gain is obtained by evaluating Equation (2) at $z = 1$, which is -3,190. Having a negative steady state gain means that free space declines as the event input rate increases, which is consistent with intuition. A second insight from Equation (2) relates to its *poles*, the values of z for which the denominator is zero. Equation (2) has a single pole at 0.985. The poles of the transfer function must lie within the unit circle of the complex plane for the system to be stable, which is the case for this transfer function. Further, poles that are closer to the unit circle indicate a system with a longer settling time (convergence time). From [9], settling time k_s is approximately

$$k_s \approx -4/ln|a|. \tag{3}$$

Applying Equation (3) to Equation (2), we determine that the open loop settling time is approximately 264 sec. That is, if there is a transitory change in the event input rate, it will take the IAP 264 sec to return to its previous state.

We design the controller with two objectives in mind. First, we want to accurately regulate free space. Second, we want to minimize the effect of *disturbances* such as changes in the types of events and the execution of administrative tasks (e.g., garbage collection) on IAP nodes. We employ proportional-integral (PI) control, an approach that is widely used because it ensures that the measured output converges to the reference input, and a PI controller is easy to understand and implement. The control law for a PI controller is:

$$u(k) = u(k-1) + (K_P + K_I)e(k) - K_P e(k-1) \tag{4}$$

where K_P and K_I are controller parameters that are determined by design.

We want the controller to settle (converge) quickly and so choose as our objective that the closed loop settling time should be 5 time units. This is achieved by properly choosing the parameters K_p and K_I. The first step is to invert Equation (3), yielding $a \approx e^{-k_s/4}$. For $k_s = 5$, $a \approx 0.449$. Next, we derive the denominator of the transfer function of the closed loop system, which is the polynomial $z^2 - (134K_P + 134K_I + 1.985)z + 134K_P + 0.985$. Setting the poles of the polynomial according to k_s and a, we get $K_P = -0.00614$ and $K_I = 0.02168$ causes this polynomial to have zeros at $0.046 \pm 0.4i$. Since $|0.046 \pm 0.4i| \approx 0.4$, the closed loop system has a settling time of approximately 5 time units.

3.2 Load Balancing Between IAPs

Load balancing provides a way to reduce response times by reducing the utilization of bottleneck resources, those resources that largely determine the response time of a system. Load balancing is particularly important in parallel computation systems involving synchronization because having an imbalance in processing speeds causes faster nodes to wait for slower nodes. In our system, the combiner is a barrier coordinator. A slow IAP forces the combiner to wait, slowing the progress of events to the second tier. The load balancer must be efficient enough to avoid being a bottleneck. It must decide where the event should go upon event arrival. It does not have time to route based on the content of the event, or hold events in a buffer for delayed decision making.

Let $\mathbf{L} = (L_1, \cdots, L_N)$ be the load in events/sec applied to the N IAPs, and let $R_1, \cdots, R_N$ be their response times at these loads. One way to formulate the objective of load balancing is to find $\mathbf{L}$ that minimizes $\sum_i (R_i - \bar{R})^2$ (where $\bar{R}$ is the average response time) subject to the constraint that $\sum_i L_i$ is constant. Unfortunately, this is a non-linear optimization that is quite complicated to solve.

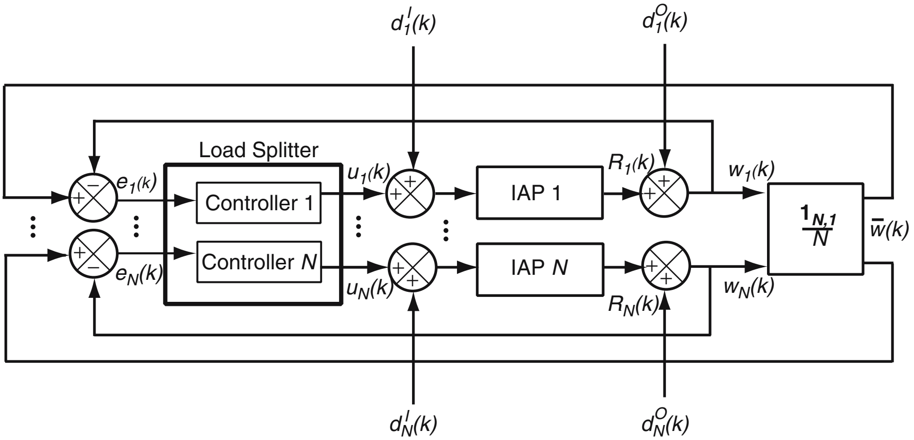


Fig. 5. Block diagram of a load balancing controller

Figure 5 depicts an approach to load balancing based on classical control theory. Employing the results in [8], load balancing is accomplished through regulatory control by having the reference input be the mean value of the (disturbance adjusted) response times of the IAPs. Such an approach allows us to regulate the IAPs so that they converge to the same value, the mean response time. In the studies we conduct in Section 4, only two IAPs are used and so the foregoing is simplified in that we need only to regulate the difference in the outputs from the two IAPs (a design that requires only one controller in the load splitter).

For small values of N, the controller can be derived in a manner similar to that done in the last section. However, for larger N, more sophistication is required. A control theory technique that is well suited to this situation is linear quadratic regulation (LQR). LQR provides a framework for constructing optimal controllers. [8] describes how to use LQR for the block diagram in Figure 5.

4 Experiments

This section describes experiments conducted to assess the control designs in Section 3.

First, we evaluate the effectiveness of the flow control system in Section 3.1 for regulating free space of the result queue. The controller is implemented as the manager of an input buffer. Every 2 sec, the controller reads the TCQ log to obtain the current size of the result queue, and uses the PI control law to calculate the number of events to send to the IAP over the next 2 seconds. Tuples that are not sent remain in the input buffer.

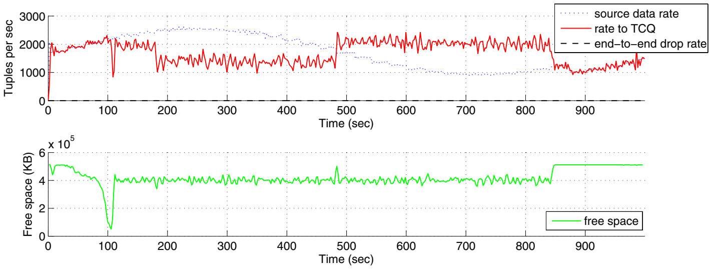


Fig. 6. Regulation of TCQ free space using a PI controller. The reference input is 400 MB. A CPU hog is introduced at time 180. The system quickly adapts the input rate so that free space returns to 400 MB.

Figure 6 plots the results of an experiment in which the reference input for the result queue is 400 MB. In the figure, the decline in free space at 100 sec is the result of a TCQ start-up effect that the controller corrects in a very short time. An input disturbance in the form of a CPU intensive application (hereafter CPU Hog) is introduced at time 160 sec, which causes a reduction in the TCQ throughput. However, when the CPU Hog completes at time 500, TCQ throughput returns to its previous level. Note that from time 100 sec through 500 sec, the input load on the TCQ is greater than its maximum capacity. Even so, the controller maintains the free space at 400 MB, and the excess load is held on the input queue for further processing, such as load balancing.

Next we assess the load balancing controller described in Section 3.2 for use as the load splitter in Figure 1. The testbed developed for this assessment is

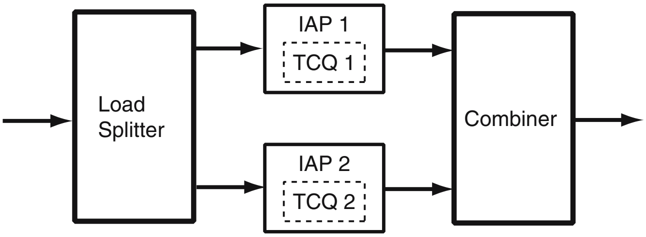


Fig. 7. Testbed used to evaluate the load balancing controller used as a load splitter. Boxes with solid lines are separate dual CPU computers with 1.5GB of RAM.

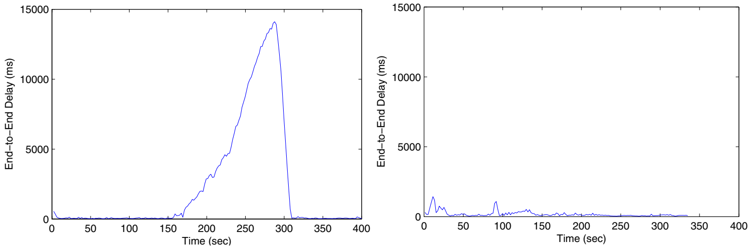


Fig. 8. LEFT: Average delay using a round-robin scheme to assign events to IAPs. RIGHT: Average delay using a well designed controller.

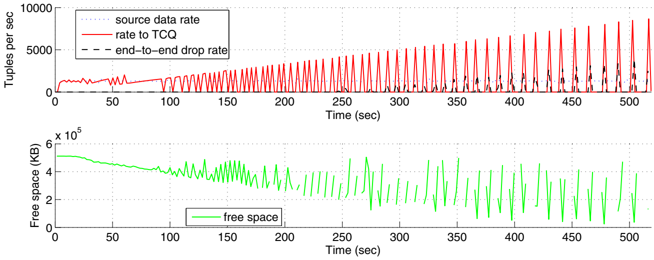


Fig. 9. Performance of an incorrectly constructed load balancing controller

depicted in Figure 7. There are two IAP nodes, and all boxes with solid lines indicate separate dual CPU computers with 1.5GB of RAM.

Figure 8 (LEFT) plots response times using a round robin scheme for the load balancing controller. This assessment includes a disturbance in the form of a CPU Hog that is started on one IAP node at time 160. We see that end-to-end delays grow to be quite large. In contrast, Figure 8 (RIGHT) plots response times when the load balancing controller is used. As in the plot on the LEFT, a CPU Hog is started on one IAP node at time 180. The load balancing controller handles this disturbance well, moderating the impact of this disturbance so effectively that there is almost no detectable change in end-to-end delays.

Last, we describe an experiment that we conducted by accident. Figure 9 plots the end-to-end delays for the system in Figure 7 using an incorrectly designed flow controller. We see that the system is *unstable* in that there are oscillations that increase in amplitude with time. At first, these characteristics seemed to be inconsistent with our control analysis, which predicted a stable system. Since control theory derives the controller from the model of the target system, we re-visited the model and discovered that it failed to consider that control actions take place in the next time interval. Correcting our models and re-designing the flow controller resulted in the performance displayed in Figure 6.

5 Conclusions

Scaling event processing requires an event processing architecture that incorporates parallelism. Intra-event processing is easily parallelized. We propose an architecture to support this parallelism in which intra-event processing elements (IAPs) are replicated to scale the system to larger event input rates. We address two challenges in this architecture. First, the IAPs are subject to overloads that require effective flow control. Second, we need to balance the load placed on processing elements to avoid resource bottlenecks. These challenges are further complicated by the presence of disturbances such as administrative tasks (e.g., garbage collection) that reduce event processing rates. We employ control theory to address both challenges since control theory provides a systematic approach to design that includes considerations of disturbances. Our solution for the flow control problem is based on regulatory control of the free space of the result queue, a key resource in our IAP implementation. Our solution for load balancing employs a technique that transforms an apparent optimization problem into a regulatory control problem. Studies done on a testbed system show that our control designs provide good performance under time varying loads and disturbances in the form of a CPU intensive application.

One area of future research is to explore the use of techniques from adaptive control and statistical learning theory to deal with stochastics and nonlinearities that are more difficult to address with classical control theory. Another direction is to expand the set of scaling experiments to gain more insight into the limitations of our current designs. Last, we want to make our IAP system available to system operators and researchers so that others can benefit from our work.

Acknowledgements

This work is in part supported by NSF grant number CNS-0509559 and Micro award number is 04-072.

References

1. L Burns, JL Hellerstein, S Ma, CS Perng, DA Rabenhorst, and D Taylor. A systematic approach to discovering correlation rules for event management. In *IEEE/IFIP Integrated Network Management*, May 2001.

2. Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 219–227, Portland, Oregon, July 2000.
3. Mike Chen, Alice Zheng, Jim Lloyd, Michael Jordan, and Eric Brewer. A statistical learning approach to failure diagnosis. In *International Conference on Autonomic Computing (ICAC-04), New York, NY*, May 2004.
4. Mike Y. Chen, Emre Kiciman, Eugene Fratkin, Armando Fox, and Eric A. Brewer. Pinpoint: Problem determination in large, dynamic internet services. In *DSN*, pages 595–604, 2002.
5. Hewlett-Packard Development Company. Hp OpenView. http://www.openview.hp.com/, 2005.
6. IBM Corporation. Tivoli. http://www.ibm.com/software/tivoli/.
7. Microsoft Corporation. Microsoft Operations Manager. http://www.microsoft.com/mom/.
8. Yixin Diao, Joseph L. Hellerstein, Adam Storm, Maheswaran Surendra, Sam Lightstone, Sujay Parekh, and Christian Garcia-Arellano. Using MIMO Linear Control for Load Balancing in Computing Systems. In *American Control Conference*, pages 2045–2050, June 2004.
9. Joseph L. Hellerstein, Yixin Diao, Sujay Parekh, and Dawn M. Tilbury. *Feedback Control of Computing Systems.* Wiley-IEEE Press, Aug 2004.
10. Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6(3):151–180, 1998.
11. Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, and Ion Stoica. Querying the internet with PIER. In *Proceedings of the 29th VLDB Conference*, 2003.
12. Sailesh Krishnamurthy, Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Samuel Madden, Frederick Reiss, and Mehul A. Shah. Telegraphcq: An architectural status report. *IEEE Data Eng. Bull.*, 26(1):11–18, 2003.
13. Robbert van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2):164–206, 2003.
14. Ricardo Vilalta, Chidanand Apté, Joseph L. Hellerstein, Sheng Ma, and Sholom M. Weiss. Predictive algorithms in the management of computer systems. *IBM Systems Journal*, 41(3):461–474, 2002.
15. S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. High speed and robust event correlation. *IEEE Communications Magazine*, 34(5):82–90, 1996.

Can Dynamic Provisioning and Rejuvenation Systems Coexist in Peace?

Raquel Lopes, Walfredo Cirne, Francisco Brasileiro, and Eduardo Colaço

Universidade Federal de Campina Grande,
Departamento de Sistemas e Computação,
Laboratório de Sistemas Distribuídos,
Av. Aprígio Veloso, 882 - 58.109-970, Campina Grande, PB, Brazil
Phone: +55 83 310 1365
{raquel, walfredo, fubica, eduardo}@dsc.ufcg.edu.br

Abstract. Dynamic provisioning systems change application capacity in order to use enough resources to accommodate current load. Rejuvenation systems detect/forecast software failures and temporarily remove one or more components of the application in order to bring them to a clean state. Up to now, these systems have been developed unaware of one another. However, many applications need to be controlled by both. In this paper we investigate whether these systems can actuate over the same application when they are not aware of each other, i.e., without coordination. We present and apply a model to study the performance of dynamic provisioning and rejuvenation systems when they actuate over the same application without coordination. Our results show that when both systems coexist application quality of service degrades in comparison with the quality of service provided when each system is acting alone. This suggests that some level of coordination must be added to maximize the benefits gained from the simultaneous use of both systems.

Keywords: Interacting systems, dynamic provisioning, rejuvenation.

1 Introduction

There are many systems that aim at automating management tasks and decisions. We are particularly interested in two of them: dynamic provisioning systems (DPS) and Rejuvenation/Restart[1] systems (RRS). DPSs have been proposed to automatically adjust the application capacity to its demand [1,2,3]. RRSs have been proposed to tackle software failures by detecting such failures and bringing the system to a clean state [4,5,6,7]. These systems have been studied separated from each other. For instance, DPSs do not take software failures nor restarts into account, while RRSs do not consider the dynamic capacity changing and the strict matching between capacity and demand provided by a DPS.

[1] Restart and rejuvenation are used here as synonyms.

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 245–256, 2005.

Nevertheless, one would expect that some applications could benefit from the use of two or more automated management systems simultaneously. In fact, in [8] we have implemented a DPS with RRS features and have obtained good results. When both features coexist application quality of service (QoS) was better and resource savings were higher. Since we built that system from scratch, it was natural to introduce some coordination between the management systems. By coordination we mean that some information can be exchanged between the DPS and the RRS in order to improve their performance. Two coordination features were implemented: (i) whenever the capacity decreases the nodes more prone to failure were selected to be removed. This information is provided for the DPS component by the RRS monitor; and (ii) instead of restarting a node, the RRS component asks the DPS component to add one more node, if possible, and then to remove the faulty one.

Ideally, instead of developing new systems, we would like to harness the potential of legacy systems independently designed. Since these systems are designed independently, they do not assume the existence of the other and, therefore, do not exchange any information. In orther words, there is no imposed coordination between their actions; they do not interact directly, but the actions of one system may interfere in the actions of the other and their actions together may influence the QoS of the managed application and its operational cost. We argue it is crucial to identify whether some imposed coordination is really needed to make these systems harmoniously coexist. This work is a first step towards better understanding the effects of having an application managed by both a DPS and an RRS that act independently.

Contributions of this paper are twofold. First, we propose a component based model to study interactions between dynamic provisioning and restart systems. Instances of this model can consider different sets of components, allowing the evaluation of different situations. Second, we instantiate different model compositions to qualitatively identify interactions between DPS and RRS that actuate independently. Our results suggest that these systems do not provide good performance when they coexist without coordination.

The remaining of this paper is organized as follows. In the next section we present a model that can be used to study interactions between DPS and RRS. In Section 3 we present an instantiation of this model. In Section 4 we analyze the results of simulations of several possible compositions of the instantiated model. Then, in Section 5 we present some related work. Finally, in Section 6, we conclude the paper and point future directions.

2 A Model to Study Interactions Between Dynamic Provisioning and Rejuvenation Systems

In order to qualitatively identify uncoordinated interactions between a DPS and an RRS, we developed a model that encompasses an application, a software error injection system (SEIS), a DPS, an RRS and a load generator system (LGS), as

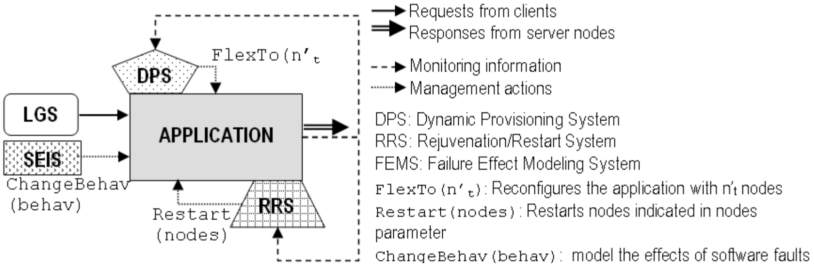

DPS
FlexTo(n'$_t$
LGS
APPLICATION
SEIS
ChangeBehav
(behav)
Restart
(nodes)
RRS
Requests from clients
Responses from server nodes
Monitoring information
Management actions
DPS: Dynamic Provisioning System
RRS: Rejuvenation/Restart System
FEMS: Failure Effect Modeling System
FlexTo(n'$_t$): Reconfigures the application with n'$_t$ nodes
Restart(nodes): Restarts nodes indicated in nodes parameter
ChangeBehav(behav): model the effects of software faults

Fig. 1. Complete model view

illustrated in Figure 1. We see this model as a component-based model, in which the application and the LGS are mandatory components.

The application is a service that runs on a cluster with load balancing such as a Web based application. Both management systems (DPS and RRS) collect monitoring information which serves as basis for management decisions. While the DPS flexes the application capacity according to its demand, the RRS detects and restarts faulty components of the application. While a node is being rejuvenated, it is not able to service requests. The SEIS changes the application behavior in order to model the effects of software faults. Finally, the LGS sends requests to the application according to some trace. Each request comes with the time required to service it (inherent service time).

2.1 The Application Model

We consider scalable applications, i.e applications whose capacity can be easily changed by changing the number of active nodes that run them. In fact, many applications satisfy this requirement; for instance, Web based applications such as e-commerce and auction sites. We consider an application with a load balancer LB and one tier with n_t active nodes at time t, as depicted in Figure 2(a). A node is active if it is expected to process requests. The LB receives requests from clients which are redirected to one of the n_t nodes following a round robin order. Requests arrival rate ($\lambda(t)$) can be highly variable.

Servers such as Apache and Tomcat run multiple processes or threads that process the requests. Each process/thread computes one request at a time. The maximum amount of processes/threads dictates the level of concurrency. The system administrator is allowed to change this number to tune the server to the amount of underneath resources. By following this idea, in each node i and every time t, there is an admission controller, composed by a pool of $m_{t,i}$ tokens. To be served, a request must first acquire a token. Requests that arrive when all tokens are in use wait in a queue called Backlog until tokens are released due to request completions. The Backlog queue has capacity K and follows a first-come, first-served (FCFS) discipline. Requests that arrive when the Backlog is full are dropped. This admission control system is illustrated in Figure 2(b).

When a request gets a token it starts to be processed. In practice, this request would be served by a network of queues composed by hardware and software

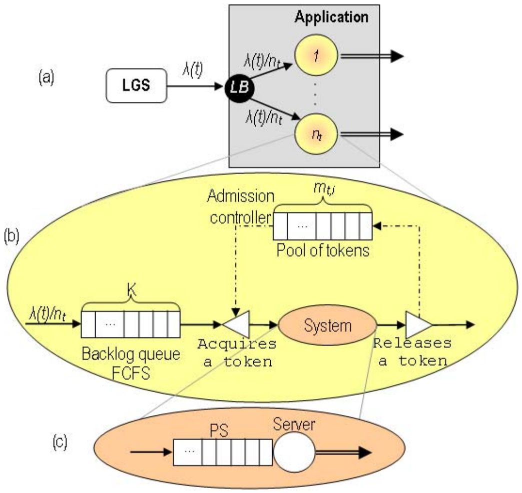


Fig. 2. Application model view

resources. We do not intend to find out what the bottlenecks of a system are, but to model the influence that the number of requests has in the system response time. Thus, we simplify the model by considering that all the admitted requests enter into a processor sharing (PS) system called Server and get equal share of system resources. If one request alone in the system is processed in t time units, when there are n requests each of them is processed in $t \times n$ time units. This system is illustrated in Figure 2(c).

2.2 The Software Error Injection System

The SEIS changes the application behavior to model the effects of software faults. We use Laprie's dependability terminology defined in [9]. He considers faults as defects that exist in the system. Faults may stay in a dormant state, or they can be activated, leading to errors. Error conditions may lead the system to a failure when the expected behavior of the system is violated.

The SEIS introduces software errors into the managed application. Such errors can lead to performance degradation or to crash/hang failures. Software errors are modeled by changing the inherent service demands that come with requests based on a particular degradation function. Errors that lead to performance degradation failures occur when the application is available but does not accomplish its expected performance. They can be modeled by incrementing inherent service demands, as we exemplify in Section 3. When inherent service demands tend to infinite we consider the occurrence of a crash/hang failure.

2.3 The Rejuvenation System

An RRS monitors each component of the application, detects/forecasts failures and restart components in order to bring them to a clean state. The granularity of these components may change from system to system. Candea *et al* [6] considers JavaBeans components, while [8,7] consider a whole process. The output of an RRS determines which nodes or components must be restarted.

2.4 The Dynamic Provisioning System

A DPS changes application capacity to accommodate the current load. It encompasses a feedback control loop. Monitoring information about the application and/or its environment is gathered and used to decide on the best number of machines to give to the application. DPSs differ from each other due to the monitoring information gathered and the objective function pursued. The output of a DPS indicates the number of nodes that must be active. Actions are needed when this number differs from the current number of active nodes. When this new value is greater than the current one, nodes must migrate from a pool of free machines to the application. Otherwise, nodes must be released. In this case, the LB immediately stops sending requests to that node, but the node remains active until all requests already in place are processed [1].

2.5 Metrics of Interest

Management systems (including the automated ones) typically aim at delivering the expected QoS at the lowest cost. Thus, an automated management system can be measured by two classes of metrics: (i) QoS metrics and (ii) cost metrics.

Two metrics related to the application QoS are going to be computed: average response time (R) and average availability (A). For response times we consider only the amount of time a request stayed in the server side. Moreover, we only consider the response times of successful responses. Availability is computed as the percentage of requests successfully processed during a measurement interval.

When a node of the data center is allocated to the application, it is in the active state; otherwise, it is in the inactive state. In order to infer the operational cost of an application we use the average number of active nodes that run the application. If we know the the amount of time the application ran and the cost of an active node per time unit we can compute the application operational cost.

3 Instantiating the Component-Based Model

In this section we present an instantiation of the generic model presented in the last section. In Subsection 3.1 we define the behavior of some components of the model and in Subsection 3.2 we present the simulation parameters used to instantiate the model compositions.

3.1 Components Behavior

The DPS aims at maintaining nodes utilization around a target, as proposed in [1]. It periodically queries the application nodes for monitoring information.

After each measurement interval the DPS queries nodes for the following measurements, computed for the previous interval: X, the number of request completions; A, the number of request arrivals; and U, the average CPU utilization (over all nodes). Given N, the current number of nodes, and ρ_{target}, the target utilization, the DPS computes the required number of servers for the next interval as follows: (i) it computes the average demand per completion as $D = U/X$; (ii) it computes the normalized utilization as $U' = max(A, X) \times D$; and (iii) it computes the number of servers needed to achieve ρ_{target} as $\lceil N' = N \times U'/\rho_{target} \rceil$.

The set of failures chosen to be modeled must be representative of the real world, since we want to uncover plausible interactions. We only consider performance degradation failures seen by other researchers. We use results of experiments carried out by Li *et al* [5] with Apache Web server. By generating a constant connection rate to the Web server they observed that response times become longer over time, degrading around 0.03 ms per hour. Based on these results we translated Li's equation into one that relates service time to the amount of requests already serviced by a node. Let $\delta_i(t)$ be the number of requests served by the node i since its last restart (or since it was activated). Each request r that arrives into a node has its service demand changed according to the following function: $S_{r,i}(t) = S_{0,r} + S_{aging}(t)$; where $S_{r,i}$ is the new service time; $S_{aging}(t) = 2.4 \times 10^{-8}\delta_i(t)$ and represents the addition due to aging in service time; and $S_{0,r}$ is the inherent service time of the request.

The RRS works as follows. After each measurement interval, the RRS gathers the current S_{aging} of each node and their availabilities. It restarts a node when: (i) the node's S_{aging} value reaches a threshold; or (ii) the node's availability is smaller than the minimum expected availability (A_{min}) for y consecutive measurement intervals. When a rejuvenation action is triggered, the RRS serves all requests already in place before restarting the node process [10].

3.2 Simulation Parameters

Both DPS and RRS are configured with 5 minutes measurement intervals. The DPS target utilization may assume three values in different simulation experiments: 65%, 75% and 85%. The RRS's S_{aging} threshold is configured as 1 second. The minimum availability (A_{min}) and y are set to 99.99% and 6 respectively.

The application parameters are set as follows. The Server system timeslice is 110 ms. The Backlog queue capacity is 1024. Both are in accordance with Linux default. Migration time and effective restart time obey a normal distribution with averages of 60 and 120 seconds, respectively. This is in accordance with some experiments we conducted using JBoss [8]. The capacities of the nodes can assume different values in different simulation experiments: around 100, 300 and 500 rps (requests per second). Nodes of low, medium and high capacity respectively have a total number of tokens ($m_{t,i}$) of 200, 500 and 1000. These numbers were chosen to avoid a request to stay more than two seconds in the Server queue.

Our simulation experiments use a 17-hour workload generated by GEIST [11]. This workload is variable and presents an average request rate of 670 rps.

The component based model allows us to build different compositions to reproduce different situations. The simplest composition is called AWoF (application without failure). The second one is called AWF (Application with failures) and is composed by the application and the SEIS. The third model is the AWF+RRS, which encompasses the application, the SEIS and the RRS. These three models use a static number of nodes calculated from results of simulations of the AWoF model. The number allows an application without failures to handle the workload without violating the 99.99% of minimum availability and the 2 seconds of maximum response times. The fourth and fifth model compositions are called AWoF+DPS and AWF+DPS. Finally, the most complete model is the AWF+DPS+RRS, which encompasses the application, the SEIS, the DPS and the RRS. In these models, DPS defines the amount of nodes to be operational on the fly. All these compositions are necessary in order to allow us to compare results of the most important model (the AWF+DPS+RRS) with results of the other intermediary models.

4 What Happens When an Application Is Controlled by a Dynamic Provisioning and a Rejuvenation System?

We ran all simulation scenarios 5 times, which resulted in 180 simulation experiments. Average application availability and response times measured for all compositions are presented in Figures 3 and 4. Average utilizations and number of nodes used are presented in Tables 1 and 2 respectively.

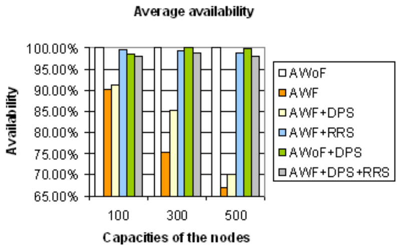


Fig. 3. Average availability

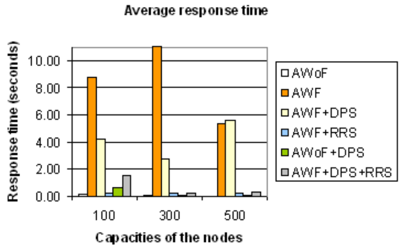


Fig. 4. Average response times

Our results show that the DPS is an efficient system. Application QoS measured for AWoF+DPS is near to the one delivered by AWoF, the model that provides the best result in terms of application QoS. However, AWoF+DPS, on average, uses 11.8% less nodes than AWoF which is statically overprovisioned. Thus, the DPS not only delivers a good application QoS but also reduces operational costs.

The RRS also proved to be efficient. When the application with failure ran without an RRS its QoS degraded, as the results of the AWF and AWF+DPS models show. These two compositions resulted the two worst QoS for the application. It is clear that the DPS is not able to manage applications with software

Table 1. Average number of active nodes

	100 rps	*300 rps*	*500 rps*
AWF+DPS+RRS	9.25	3.64	2.39
AWoF+DPS	9.19	3.55	2.25
AWoF, AWF, AWF+RRS	10	4	3
AWF+DPS	9.46	3.76	2.19

Table 2. Average utilization

	100 rps	*300 rps*	*500 rps*
AWF+DPS+RRS	70.1%	63.6%	59.9%
AWoF+DPS	70.2%	63.2%	59.6%
AWoF	66.2%	55.6%	44.8%
AWF+RRS	66.9%	57.8%	49.5%

faults. When the RRS runs (AWF+RRS composition) application QoS increased and was near that of AWoF.

Although DPS and RRS perform very well in isolation, application QoS degrades when they coexist. For nodes of low, medium and high capacities, application availability of the AWF+DPS+RRS model is respectively 0.59%, 0.30% and 0.73% worse than the minimum availability between the ones measured for AWoF+DPS and AWF+RRS. At a first glance, availability does not seem to vary substantially. However, small variations in availability represent big differences in terms of number of requests served. With a request arrival of 670 rps, an availability loss of 0.30% leads the application to drop almost 180,000 more requests per day. For nodes of low, medium and high capacities, response times of the AWF+DPS+RRS model are 60.4%, 30,8% and 17.9% higher than the the maximum response times between the ones measured for AWoF+DPS and AWF+RRS.

By comparing the number of nodes used by AWoF+DPS and AWF+DPS-+RRS we find out if the DPS changes its decisions due to the RRS actuation or aging. On average, the number of nodes used by AWF+DPS+RRS is greater than the number of nodes used by AWoF+DPS (Table 1): 0.6%, 2.5% and 5.9% greater for nodes of low, medium and high capacities. The AWF+DPS+RRS composition uses more nodes than the AWoF+DPS one because of the way failures are modeled. The SEIS models performance degradation faults. Requests in a faulty node require more time to be served. This failure presents two consequences. First, response times are greater, since service times are greater. Second, faulty nodes present greater utilizations, because requests stay longer in the system. Since the DPS goal is to maintain nodes' utilizations around a target, it adds more nodes when there are faulty nodes. Besides, when a node is selected for rejuvenation the DPS is influenced even more and augments the amount of nodes used. Nodes are added during or immediately after rejuvenation, probably to suppress the lack caused by the node being restarted, and are removed shortly after being added. During a restart, the difference in terms of number of nodes used by AWoF+DPS+RRS and AWoF+DPS increases to 1.6%, 12.5% and 27.7% for nodes of low, medium and high capacities respectively (Table 3).

Let us now discuss what happens with the application QoS during the restarts. We present average application availability and response time during restarts in Figures 5 and 6. The average number of nodes used and their utilizations during restarts are presented in Tables 3 and 4 respectively.

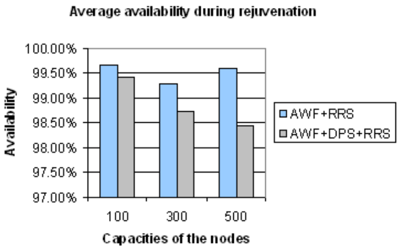


Fig. 5. Average availability (restart)

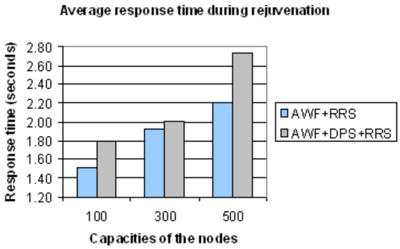


Fig. 6. Average response times (restart)

Table 3. Average number of active nodes during rejuvenation

	100 rps	*300 rps*	*500 rps*
AWF+DPS+RRS	8.94	3.91	2.92
AWoF+DPS	8.80	3.42	2.11
AWF+RRS	10	4	3

Table 4. Average utilization during rejuvenation

	100 rps	*300 rps*	*500 rps*
AWF+DPS+RRS	74.3%	77.2%	68.3
AWoF+DPS	71.8%	65.3%	60.2%
AWF+RRS	72.0%	72.7%	62.8%

Overprovisioned applications cope better with rejuvenation than the applications managed by a DPS. We found out that, during restarts, for low, medium and high capacity nodes respectively, average application availability measured for the AWF+RRS is 0.25%, 0.56% and 1.16% better than the availability measured for the AWF+DPS+RRS composition. The application response time during rejuvenation of nodes of low, medium and high capacities is 16.1%, 4.0% and 19.3% better for the AWF+RRS model than for the AWF+DPS+RRS one.

When both systems coexist the average utilization of the nodes during restart is greater than the utilization of the nodes from the AWF+RRS (Table 4). The overprovisioned AWF+RRS composition always uses more nodes than the AWF+DPS+RRS one. The number of nodes used by the DPS during the restart of a node for the AWF+DPS+RRS model is not enough to suppress the lack of the node being restarted. As a result, the active nodes of the AWF+DPS+RRS model become more saturated during restarts, increasing the probability of request rejection (when the Backlog is full) and increasing response times.

The rejuvenation time for the AWF+DPS+RRS composition is 0.33, 0.40 and 0.56 hours for nodes of low, medium and high capacity respectively. For the AWF+RRS model, the average restart time is almost equal those ones: 0.34, 0.41 and 0.55 hours for nodes of low, medium and high capacity respectively. Thus, the AWF+DPS+RRS composition spends, on average, the same time rejuvenating nodes, however, during these moments, application QoS degrades more when both DPS and RRS coexist then for the AWF+RRS composition.

To sum up, our results suggest that when both systems coexist application QoS may degrade in comparison with the QoS provided when each system is acting alone. This is an indicative that they are not orthogonal systems, in the sense that they are not independent. We believe that some level of coordination must be added to maximize the benefits gained from the simultaneous use of both systems. In fact, our previous experience [8] shows that some coordination between DPS and RRS can provide good results.

5 Related Work

The dissemination of overlay networks over IP (Internet Protocol) networks results in two independent systems coordinating data routing. In [12] interactions between these systems are studied. When failures occur, these interactions interfere in some traffic engineering tasks and, when there are overlay networks that span different autonomous systems, they allow the network status of a system to influence the network status of others, which is undesirable. We here also investigate interactions between systems that act over the same target.

Systems with conflicting goals is presented in [13]. Some conflicting relationships arise when a complex application presents both real time and fault tolerance requirements. In fact, some middlewares offer real time guarantees and others offer fault tolerance behavior. However, when real time requirements coexist with fault tolerance ones, the simple union of systems exclusively designed to deal with individual cases is not enough [13,14]. DPS and RRS systems have sometimes conflicting goals. For instance, RRS can restart a saturated node that was delivering low availability during a load surge.

Graupner *et al* [15] alert to interactions among traditional management systems and virtualization management systems, which recently arose as separated management systems. Associations among applications and underneath resources change more often in virtualized environments, under the control of the virtualization layer. If no information is exchanged between the traditional management system and the virtualization system, the traditional management system becomes unaware of these dynamic associations. It is difficult to separate the virtualization and the traditional management and these activities should be made in a combined way. This vision of various management systems which interact among them is exactly the vision we consider here. Interactions between these systems must be known in order to allow them to coexist synergistically.

Finally, control theory researchers study how independent control systems acting under the same plant (controlled system) and in the presence of uncertainties actuate such as the desired global state of the system is reached. This problem is known as the decentralized adaptive control problem [16]. According to [16,17,18] distributed control systems need to exchange information in order to reduce the error of control actions. In [17], for instance, control systems actuate in different and pre-scheduled moments and when one system makes a decision about a control action the others must be aware of that. In [18] the control system which actuate over a subsystem must know the desired status of all subsystems that compounds the plant. It is clear that an extra effort is needed to coordinate the actions of control systems which actuate over the same plant. In the future, similar techniques may be introduced into DPS and RRS behavior in order to make their coexistence more efficient. This work provides some insights on how these systems coexist without these techniques.

6 Conclusions and Future Research

In this paper we propose a component based model used to better understand the interactions between dynamic provisioning and restart systems that act without coordination over the same application. We implemented the model and instantiated many different compositions. Then, we evaluated the model through simulation experiments.

Even in the very simple scenario studied, where only one node can be rejuvenated at a time, we can see that DPS and RRS may interact in a way that application QoS is degraded when both systems run simultaneously if compared to performance of systems that use each of the management services independently. Even worse, performance degrades when a slightly great number of resources are used. We believe that for more complex applications, with many tiers, and higher failure rates these unwanted interactions will be even more present.

According to our results, it is inefficient to join a DPS and an RRS that are not aware of each other. We believe they can exchange some information to allow the coordination of their actions in a synergistic manner. The way the application fail and the moments when one or more nodes are restarted influence not only the application QoS but also DPS actions. If DPS is aware of failure information and RRS actions it could act in a more efficient manner. One of these positive manners was presented in [8], where aged nodes were released when a capacity decreasing was performed.

Our next step is to propose an coordination module that makes the coexistence of these systems completely harmonic. This additional module is interesting because it allows us to harness mature dynamic provisioning and rejuvenation systems. We plan to analyze and validate this coordination module by carrying out simulations and measurement experiments.

Acknowledgments. We would like to thank Dr. Alexander Keller and the anonymous reviewers for their helpful comments. This work was developed in collaboration with HP Brazil R&D and funded by CNPq/Brazil grants 141655/2002-0, 302317/2003-1 and 300646/1996-8.

References

1. Ranjan, S., Rolia, J., Fu, H., Knightly, E.: Qos-driven server migration for internet data centers. In: Proceedings of the International Workshop on Quality of Service. (2002) 3 12
2. Lassettre, E., et al: Dynamic surge protection: An approach to handling unexpected workload surges with resource actions that have dead times. In: 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management. Volume 2867 of Lecture Notes in Computer Science., Springer (2003) 8292
3. Urgaonkar, B., Shenoy, P.: Cataclysm: Handling extreme overloads in internet applications. In: Proceedings of the Fourteenth International World Wide Web Conference (WWW 2005). (2005)

4. Garg, S., et al: Analysis of preventive maintenance in transactions based software systems. IEEE Transactions on Computers 47 (1998)
5. Li, L., Vaidyanathan, K., Trivedi, K.S.: An approach for estimation of software aging in a web server. In: International Symposium on Empirical Software Engineering. (2002)
6. Candea, G., et al: Microreboot a technique for cheap recovery. In: Proceedings of the 6th Symposium on Operating Systems Design and Implementation. (2004)
7. Hong, Y., Chen, D., Li, L., Trivedi, K.: Closed loop design for software rejuvenation. In: Workshop on Self-Healing, Adaptive, and Self-Managed Systems. (2002)
8. Lopes, R., Cirne, W., Brasileiro, F.: Improving dynamic provisioning systems using software restarts. In: Fifteenth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management. LNCS. Volume 3278., Springer (2004)
9. Anderson, T., ed.: Edpendability of Resilient Computers. Blackwell Scientific Publications, Oxford (1989)
10. Candea, G., Fox, A.: Recursive restartability: Turning the reboot sledgehammer into a scalpel. In: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems. (2001) 125132
11. Kant, K., Tewari, V., Iyer, R.: Geist: A generator of e-commerce and internet server traffic. In: Proceedings of the 2001 IEEE International Symposium on Performance Analysis of Systems and Software, IEEE Computer Society (2001) 4956
12. Keralapura, R., Taft, N., Iannaccone, C.N.C.G.: Can isps take the heat from overlay networks? In: ACM SIGCOMM Workshop on Hot Topics in Networks. (2004)
13. Narasimhan, P.: Trade-offs between real-time and fault tolerance for middleware applications. In: Workshop on Foundations of Middleware Technologies. (2002)
14. Stankovic, J.A., F.Wang: The integration of scheduling and fault tolerance in real-time systems. Technical report, UM-CS-1992-049, Department of Computer Science, University of Massachusetts (1992)
15. Graupner, S., et al: Impact of virtualization on management systems. Technical report, Hewlett-Packard Laboratories (2003)
16. Mukhopadhyay, S.: Distributed control and distributed computing. SIGAPP Appl. Comput. Rev. 7 (1999) 2324
17. Mukhopadhyay, S., Narendra, K.S.: Decentralized adaptive control using partial information. In: American Control Conference. Volume 1. (1999) 34 38
18. Narendra, K.S., Oleng, N.O.: Decentralized adaptive control. In: American Control Conference. Volume 5. (2002) 3407 3412

A Hierarchical Architecture for a Distributed Management of P2P Networks and Services

Guillaume Doyen, Emmanuel Nataf, and Olivier Festor

The Madynes Research Team, Loria,
615, rue du Jardin Botanique,
54602 Villers-lès-Nancy, France
{doyen, nataf, festor}@loria.fr

Abstract. We propose a management architecture for the P2P model which respects its distributed nature while building a hierarchical structure. This architecture enables the distribution of management functions, avoids an excessive centralization of the manager role and fits the dynamic of the P2P model well. The architecture is evaluated through an implementation in the Pastry framework.

1 Introduction

Nowadays, P2P networking is an emerging model that extends the limits of the client/server approach. Indeed, applications built on top of it present better scalability, load balancing and fault tolerance. Enterprises, administrations or universities are interested in the deployement of P2P applications for purposes like the distribution of networked file systems, including data replication mechanisms, or the use of distributed collaboration tools for projects that count remote participants. Network and service providers also see a good opportunity in supporting P2P applications with service level agreements. In this context, the need for a management framework for these services is obvious in order to ensure service levels for value-added applications.

The power of the P2P model relies on the distribution of all resources, knowledge and load. We believe that the management of a P2P community cannot be achieved in a centralized way mainly because such a centralization can potentially strongly affect the advantages brought by the P2P model. It does not make sense for a P2P community to have a central authority which manages all the peers: all the efforts done to increase the service level by the use of a distributed model will be impacted by the addition of a centralized framework, which actually owns the same goal of service operating improvement. This is why, in the same way peers act both as client and server, they have to act both as manager and agents for their management plane. Thus, the management of P2P services should be achieved through a P2P approach and, in this paper, we present a framework which takes the advantages of both the P2P model for management task distribution and the centralized management approach with the use of the standard manager and agent roles, to build a hierarchical management architecture for P2P networks and services. This architecture fits the

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 257–268, 2005.

P2P model characteristics, which are decentralization, dynamic of peers naming and presence, heterogeneous nature of involved devices, and behavior of participants. Moreover, it presents interesting properties concerning the load-control of manager nodes, the structure balance and the choice of nodes for crucial points.

The paper is organized as follows: motivations are given in section 2. section 3 deals with the current research works that address the management of P2P networks and services. Work on tree-based infrastructures for P2P networks is also addressed in this section. Section 4 presents the objectives we want to reach through our proposal and the general algorithm we designed for the tree construction. The way we distribute this algorithm among peers is shown in section 5 and deployment aspects are treated in section 6. Finally, some conclusions and directions for future works are given in section 7.

2 Motivation

The proposal of a hierarchical distributed management model that is aligned with the underlying P2P framework it manages is driven by the following motivations:

Resist to scale: P2P infrastructures involve a large number of components often spread among multiple administrative domains. Only self-management capabilities built in these complex infrastructures can provide scalable and efficient management;

Master the dynamics: The individual components of a large P2P infrastructure are expected to be very dynamic (i.e. versatile presence and contribution to a service). Traditionnal management systems in which all participating components and ressources are known in advance and registered cannot be applied there.

Our approach is based on a partial integration of the management plane in the service plane of the P2P infrastructure. Such an integration avoids developers and service operators to have to deal with two different worlds (naming schemes, access protocols, security issues, ...). JMX is a good example of such successful integration in the Java world. Moreover, the merging of dedicated management signaling with the existing inftrastructure signaling potentially reduces the management overhead.

Our architecture is hierarchical since it has proven efficient for many monitoring operations, i.e. those based on monotonic functions (e.g. *Sum*, *Min*, *Max*, *Count* and *Average*) [1]. It is also very well adapted to the dynamics of the underlying environment.

3 Related Work

3.1 P2P Management

Currently, a lot of applications, built over different protocols, allow users of a community to share files. Besides the fact that shared data are copyrighted, the major problem content sharing applications have to face concerns the free riding which consists, for a peer, in the use of other peers' resources without providing any themselves [2]. This phenomenon clearly shows the need for a management framework able to ensure service levels. From this point, several proposals have emerged. They are service-embedded and use incentive approaches which rely on economic models [3]. For example, the MMAPPS (Market Management of Peer-to-peer Services) project proposes a cost evaluation for resources that depends on their availability, interest and quality [4].

Concerning performance management, [5] proposes to use an active network framework dedicated to Gnutella-like applications. First, it enables the scattering of a community into sub domains, thus limiting the scope of messages which rely on a flooding method. Then, messages routing adapts itself to the traffic load between peers. Thirdly, the virtual topology is adapted to the physical underlying network, which increases the global overlay performance. This work presents interesting results and is deployed over a Gnutella like infrastructure.

The major work concerning the deployment of a management infrastructure for P2P networks and services concerns the MMP[1] project of Jxta [6]. Jxta is a generic platform for the development of P2P services. From a functional perspective, Jxta provides an abstraction of basic P2P mechanisms like routing, lookup, organization or communication. It makes the development of services easier and allows their interoperability. The MMP project aims at providing a management infrastructure for Jxta communities. To do that, it provides an instrumentation of Jxta peers, a remote monitoring service and a management console application. The idea of this work is very interesting but actually, the instrumentation of Jxta peers is incomplete and the MMP project is now abandoned.

3.2 Our Previous Work

One of our goals is to design and deploy a management infrastructure which can be independent from the underlying services and which relies on standard approaches of network management. A first instrumentation experiment of a instant messaging P2P application clearly expressed the need for a management framework for such a class of application [7]. Then, we designed a generic management information model for P2P networks and services [8]. The latter enables a manager to build an abstract view of a P2P community, participating peers, shared resources and deployed services. We used CIM (Common Information Model) [9] as a formalism to express our model. In a second step, we refined our model towards the performance management of DHTs[2] [10]. As a case study, we considered Chord [11] and we defined a set of metrics which feature the performance of this DHT. Then, we integrated these metrics into our model. By this way, we enable a manager to evaluate the global performance of a Chord ring.

Our current work concerns the architectural aspects for the management of P2P networks and services. We are working on a proposal for a management

[1] Metering and Monitoring Project - meter.jxta.org.

[2] Distributed Hash Tables.

architecture that is compliant with the characteristics of the P2P model, and we present it in this paper.

3.3 Existing Overlay Tree Proposals

We propose to use a tree structure to enable a root manager to aggregate management information provided by sub-manager and agents in order to build an abstract view of a P2P community. Nevertheless, there are many other use cases where building a tree structure is required.

El-Ansary et al. [12] propose the building of a broadcast tree for structured P2P networks. Despites it presents interesting properties, their algorithm is strongly dependent on specific components of Chord [11].

From a theoretical perspective, our approach presents many similarities with [13]. The authors propose to use a binary tree structure to build a DHT. Their building principle and simulations provide very interesting results, such as the cost for node insertion or removal which falls from $log^2(N)$, in [11,14] to $log(N)$. For management purpose however, the use of a binary tree is not the best choice, mainly because the tree is too thin and deep and this can be problematic, for instance, to propagate alarms from a leaf to the root.

Current work which our work is the most closest to is proposed in [15]. Its objective is to build an aggregation tree over any DHT that enables the computation of aggregation functions. The definition of a *Parent* function allows any node to establish a link towards its parent in the tree. Such a function has to be well chosen, so that it ensures a good tree balance. Our work is very similar, but it achieves a broader objective in the sense that if our management tree enables the computation of aggregation functions, in a more general way, it defines the roles of manager and agent for nodes.

4 Foundations and Principle

4.1 Goal

There are several objectives we want our management architecture to reach. These are:

Optimal manager role distribution: The P2P model is a distributed model where there is potentially no central point; each peer acts as both a client and a server. From a management perspective, we want to distribute the manager role among most peers so that they act as agent and manager.

Structure balance: To fit the distributed aspect of the P2P model, we want our tree to be well balanced so that a node cannot act as an excessive central point of failure and be stressed more than others.

Manager election: The more managers are close to the tree root, the more their role is crucial to achieve management functions. This is why we want to be able to choose managers according to any application context criteria such as the hardware resources or the user behavior.

Depth constraint: To ensure a minimum performance level of the management architecture, we want to control the tree depth so that a manager can contact any agent in a controlled number of hops as well as an agent, located as a leaf to contact the root manager.

4.2 General Tree Construction Principle

In order to build our hierarchical structure, we define the following axioms:

1. Each node is an agent and eventually a manager (at most once);
2. Each leaf represents an agent;
3. Each intermediate node, up to the root, represents a manager;
4. Each node owns an identifier which is the one given at the DHT level.
5. Each node owns a metric, called $Weight$, which represents the quality of the node for the manager role. This metric is based on any relevant criteria such as the hardware capabilities, the behavior or the participation level of the node. It used to choose the managers so that nodes with the highest weight are the managers of the highest levels in the hierarchy.

Then, we define that: *each manager of level L is responsible for nodes of level $L + 1$ that present a common prefix of L digits.* Moreover, *managers are chosen through an election process.* The construction principle used here is very similar to the one proposed in [16].

Figure 1 shows a simple tree example applied to a Pastry-like DHT. One can see that each leaf represents an agent and appears zero or one time in upper levels where the managers are placed. Moreover, each manager owns a common prefix with its children that depends on its level. For example, node 001 located at level 2 presents the common prefix 00 with each of its agents that are 001, 002 and 003.

4.3 Formal Definition

Our structure follows a formal definition that is expressed using a first order logic statement. Consider the following construction parameters:

B	The identifiers' base
D	The number of identifier's digits
N	The set of nodes in the community

Then, we define the following sets and variables:

d_i	The digit of rank i of an identifier with $0 \leq d_i < B$ and $1 \leq i \leq D$
$d_1 \ldots d_D$	A node identifier
L	The number of digits of a prefix
λ	The set of all levels present, that is the set of all L
$Q_{d_1 \ldots d_L}$	The set of nodes that own the prefix $d_1 \ldots d_L$
P_L	The set of set of nodes $Q_{d_1 \ldots d_L}$ which owns a common prefix of L digits
G	The set of manager nodes.

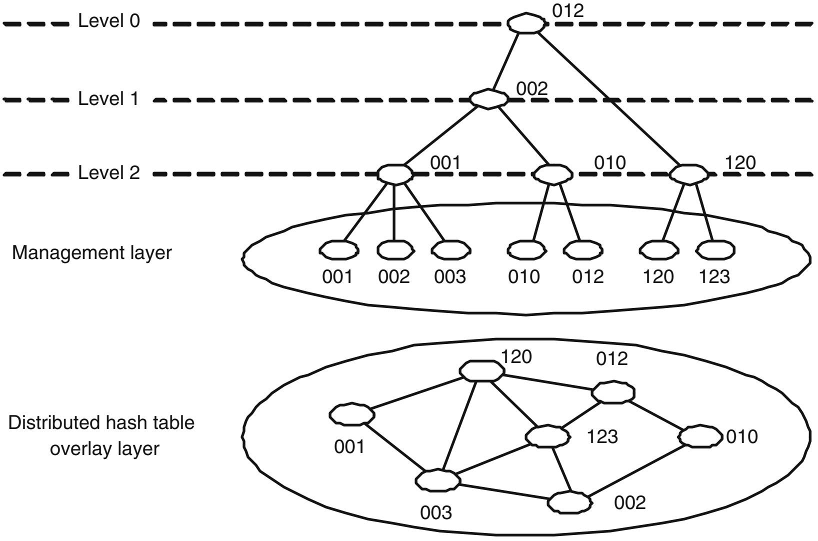


Fig. 1. Management tree example applied to a DHT

The statement below is always true with nodes that are involved in any management process. As we will see in the next section, only arriving nodes or temporarily disconnected nodes are not involved in management functions.

```
Tree definition ()
1   $P_{-1} \leftarrow \{\emptyset\}$
2   $\forall L \in \lambda$
3      $\forall n \in N$
4         $Q_{d_1 \ldots d_L} \leftarrow Q_{d_1 \ldots d_L} \cup \{n \mid n.\text{Prefix}(L) = d_1 \ldots d_L\}$
5      $P_L \leftarrow \{Q_{d_1 \ldots d_L} \mid Q_{d_1 \ldots d_L} \neq \{\emptyset\}, 0 \leq d_i < B, 1 \leq i \leq L\}$
6      $\forall P \in P_L \setminus \{P_L \cap P_{L-1}\}$
7         $G \leftarrow G \cup \{n \mid n \in P, n \notin G, n.\text{Weight}() = max(p.\text{Weight}(), p \in P)\}$
```

where $n.Prefix(L)$ returns a list of L former digits from the node n identifier and $n.Weight()$ gives the node's quality according the metric defined in section 4.2.

In line 4, we build $Q_{d_1 \ldots d_L}$ the sets of nodes that own a common prefix of L digits. Then, in line 5 we gather all the non empty $Q_{d_1 \ldots d_L}$ sets in the P_L set of sets. Finally, in line 7, if needed, we elect a free manager which presents the highest weight. This way, we construct a tree that fulfills the goals presented in section 4.1.

5 Distribution of the Algorithm

We have designed two protocols for each event that can occur in the life of a P2P community. These events are: the arrival of a new node requiring an attachment to the community and the departure of a node. Together with them, a regular maintenance process triggers structure update when the above mentioned events occur.

5.1 Node Insertion

The insertion process aims at adding a new node in the structure at its right location. It consists in looking for the manager that owns the longer prefix with the arriving node. In current DHTs, such an operation is not trivial since DHT functions do not enable semantic lookup. One solution could rely on a method of successive approaches: an arriving node looks for a node that owns a $D - 1$ prefix, then a $D - 2$ one, until it finds a manager. But, the method is costly in term of number of messages.

To overcome the above constraints, we propose to use the following method: when a node joins the tree, it generates a request with a random identifier. According to the DHT properties, a node with an identifier close to the required one responds. Then, the arriving node requests the manager of the latter node for its management. Thus, the new node is inserted, but the tree is not consistent regarding our formal description. This is not a problem because the new node will be involved, as either agent or manager, until it is correctly placed in the management architecture through the maintenance process.

5.2 Maintenance

The maintenance process is executed by manager nodes. It aims at maintaining the structure consistent. It is composed of two functions: the first consists in enforcing that the tree construction rules are effectively applied, and the second consists in verifying that referenced nodes are still alive.

This is why the maintenance process is executed in several contexts: (1) when a manager detects a new node insertion, to check that the arriving node is well located, and (2) at regular intervals, to check for any node departure.

The different operations executed by the maintenance process are:

1. **Presence checking:** For an L level manager, it consists in checking the presence of each of its children and its father. To do that, maintenance requests are sent to each child. Whenever a child doesn't respond, it is removed from the children list. Moreover whenever no maintenance request is received from its father, the manager is considered orphan and restarts the join process.
2. **Prefix checking:** For an L level manager, it consists in checking the children prefixes consistency. Two cases of reorganization are possible:
   - **Too short prefixes:** Whenever a child doesn't own a prefix of L digits with the considered manager, it transfers the child to its father;
   - **Longer prefixes:** Whenever two or more children share a prefix longer than L, two cases are possible:
     - **Agent and manager children:** If agents and managers share a common prefix longer than L, then the manager with the highest weight will manage the other ones.

- **Identical children:** If children that share a longer prefix are exclusively a set of managers or agents, the child of higher weight will manage the other ones.

3. **Weight checking:** For an L level manager, it consists in checking that it does not reference any child, that is not a manager, and that owns a higher weight than its own weight. Each time this case occurs, the child of highest rank will take the place of the current manager and the latter loses its manager role.

5.3 Node Departure

When a node leaves the management structure and informs its father, all its children will be managed temporarily by the father, until the maintenance process reorganizes the structure. In case a node leaves the tree without informing its father, the maintenance process of neighbor nodes will detect its absence. The father will detect the absence of response from one of its children, and the children of the failing node will detect that they have not received any maintenance message for a given time. These orphan nodes will therefore use the insertion process to join the tree again.

6 Deployment

We have designed a prototype implementation of our architecture. It is built over the Java Free-Pastry[3] implementation of Pastry [14]. In this section, we first detail some implementation aspects. Then, we present the tests we have performed and the results they provide.

6.1 Node Architecture

The code we have deployed follows the functionnal architecture shown on Figure 2. Each Pastry node is composed of two different parts: an agent and a manager. The manager part is activated if the node endorses a manager role.

The agent part is composed of two main entities. The first one is a JMX MBean server. It hosts standard MBean objects coming from the instrumentation of Pastry presented below. The second entity of the agent part concerns core agent functionalities (requests processing, father soft state maintenance, . . .).

The manager part is, as for the agent part, composed of two entities: the manager core and the maintenance process. The core entity is in charge of standard management functions of the tree, like requests forwarding or partial computations of a management function. The maintenance process is responsible for ensuring the consistency of all meta-data stored in the state manager, represented in the upper part of Figure 2. This process periodically executes the operations described in section 5.2.

[3] freepastry.rice.edu

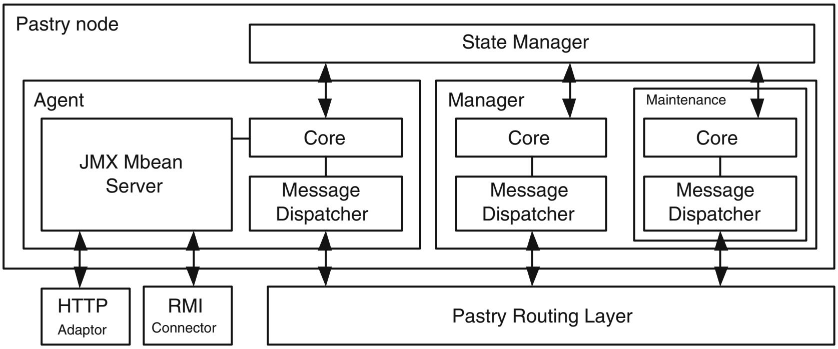


Fig. 2. Node architecture

Concerning the communication, all the tree construction and maintenance related messages are exchanged through the Pastry routing layer. Management access to JMX MBean servers is achieved through a RMI as defined in JMX.

6.2 Node Instrumentation

To design a manageable DHT community, we have instrumented Pastry nodes. Managed objects are CIM instances which follow the information model proposed in [10]; we collect information concerning routing tables, leaf sets and lookup and maintenance services. In fact, our management plane addresses two levels: a local one where managed objects stand for data related to their host node, and a global level, addressed by a manager (which can be centralized, hierarchical or distributed) which aggregates local information to provide an abstract global view of a community.

All the local and global managed objects are registered into a JMX MBean server as standard MBeans and we use RMI to enable the communication between these entities. To validate this instrumentation, we have designed a small application which draws a topological view of a managed Pastry community. We have chosen to use the leaf associations as a topological criterion because it respects the neighborhood semantic.

6.3 Evaluation

We present here the results of a small scale test that evaluates the construction cost of our management architecture according to the number of nodes. We have considered scenarios which involve from 1 to 20 nodes. Identifiers of nodes are set randomly using the Pastry factory. The nodes' weight, represented as a byte value, is chosen randomly. Concerning timing aspects, the node arrival rate has been fixed to 1 node per minute. The maintenance process is executed every 15 seconds and a timeout for messages has been set to 30 seconds.

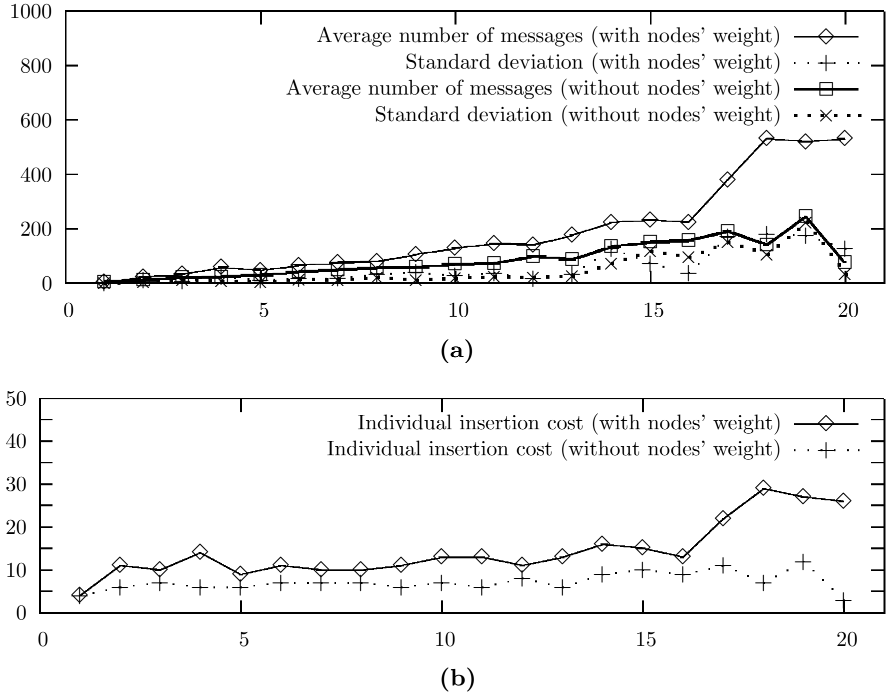


Fig. 3. (a) Evaluation of the global tree construction cost. (b) Evaluation of the individual insertion cost.

We did perform two tests. The first one considers the nodes' weight while the second one doesn't, i.e. step 3 of the maintenance process is executed in the first case only. Figure 3.a depicts the global construction cost. The metric we have considered is the number of messages exchanged between nodes to build the tree. For each of the two tests, we have represented the average value and the standard deviation. Figure 3.b represents the insertion cost for one node expressed in term of the number of messages exchanged. On this Figure, we have represented the average value of this metric for the two tests.

When considering the nodes' weight, one can see that from 1 to 16 nodes, the individual insertion cost is constant with a mean value of 12 messages per node. From 16 nodes, the insertion cost doubles. This phenomenon is due to the fact that statistically, up to 16 nodes with random identifiers, the tree contains only one level: a root manager in charge of agents; but from 16 nodes, the tree tends to present a second level; this is why the insertion cost increases. Then from 16 nodes, the standard deviation increases, because, the more nodes join the tree, the more different scenarios leading to different tree construction operating occur.

Concerning the second test, Figure 3.a and 3.b show that the tree construction cost is lower when the structure does not consider the nodes' weight. Moreover, the evolution of the construction cost is more regular than in the first test.

Finally, one can remark that, in the latter case, the two cases which count 18 and 20 nodes are not statistically correct and show that the more nodes there are in the community, the more tests we have to perform to obtain meaningful results.

To conclude, this test shows that the consideration of a weight metric increases the construction cost strongly and may be removed to improve the performance of the tree structure.

7 Conclusion and Future Works

In this paper, we expressed the need for a management framework for P2P network and services; a management infrastructure is essential to enable a common use of the P2P model in sensitive value-added services. We proposed a hierarchical management architecture that fits the P2P model characteristics and that relies on the naming properties of peers. Our structure enables an strong distribution of the manager role, provides a balanced structure and a tree depth control. Moreover, the addition of a weight metric to peers ensure that critical places will be used by best participants. To implement our model, we have proposed a distributed algorithm which consists of three processes: insertion, removal and maintenance, responsible for enforcing the structure consistency.

Free Pastry was used to implement our model . We have instrumented nodes and integrated managed objects into a JMX MBean server. A prototype has been deployed and the tree structure building algorithm validated.

We plan to test our architecture in a cluster of five hundred nodes. In this context, we will be able to perform tests for communities containing up to 10000 virtual nodes and check the scalability of our proposal. Future tests will address (1) the tree resistance to nodes failures and (2) the way we can tune the prefix consideration to reach particular management objectives; for example, a management infrastructure which deals with fault management and alarm propagations requires a very short tree depth but the manager nodes will be strongly loaded. Besides this case, applications which want to spread management functions among peers will require a deep tree involving as most peers as possible.

References

1. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tag: a tiny aggregation service for ad-hoc sensor networks. SIGOPS Operating System Review **36** (2002) 131–146
2. Saroiu, S., Gummadi, P.K., Gribble, S.: A measurement study of peer-to-peer file sharing systems. In: Proceedings of Multimedia Computing and Networking 2002 (MMCN ’02), San Jose, CA, USA (2002)
3. Hellerstein, J.L.: A comparison of techniques for diagnosing performance problems in information systems (extended abstract). SIGMETRICS Perform. Eval. Rev. **22** (1994) 278–279
4. Antoniadis, P., Courcoubetis, C.: Market models for p2p content distribution. AP2PC’02, Bologna, Italy (2002)

5. deMeer, H., Tutschuku, K.: A performance management architecture for peer-to-peer services based on application-level active networks. In Stadler, R., Ulema, M., eds.: Networks operations and management symposium (NOMS 2002), IEEE (2002) 927–929
6. Oaks, S., Traversat, B., Gong, L.: Jxta in a nutshell. O'Reilly (2002)
7. Doyen, G., Festor, O., Nataf, E.: Management of peer-to-peer services applied to instant messaging. In Marshall, A., Agoulmine, N., eds.: Management of Multimedia Networks and Services. Number 2839 in LNCS (2003) 449–461 End-to-End Monitoring Workshop 2003 (E2EMON'03).
8. Doyen, G., Festor, O., Nataf, E.: A cim extension for peer-to-peer network and service management. In De Souza, J., Dini, P., eds.: Proceedings of the 11th International Conference on Telecommunication (ICT'2004). Number 3124 in LNCS (2004) 801–810
9. Bumpus, W., Sweitzer, J.W., Thompson, P., R., W.A., Williams, R.C.: Common Information Model. Wiley (2000)
10. Doyen, G., Nataf, E., Festor, O.: Performance management of distributed hash tables. In: To appear in The European summer School (EUNICE'2005). (2005)
11. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: 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, ACM Press (2001) 149–160
12. El-Ansary, S., Alima, L.O., Brand, P., Haridi, S.: Efficient broadcast in structured p2p networks. In: 2nd International Workshop on Peer-to-Peer Systems - IPTPS '2003. (2003) 304–314
13. Buccafurri, F., Lax, G.: Tls: A tree-based dht lookup service for highly dynamic networks. In: CoopIS, DOA, and ODBASE. Number 3290 in LNCS (2004) 563–580
14. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware). (2001) 329–350
15. Li, J., Lim, D.Y.: A robust aggregation tree on distributed hash tables. In Sinha, V., Eisenstein, J., Sezgin, T.M., eds.: Proceedings of the 2004 Student Oxygen Workshop. (2004)
16. Plaxton, C.G., Rajaraman, R., W., R.A.: Accessing nearby copies of replicated objects in a distributed environment. In: ACM Symposium on Parallel Algorithms and Architectures. (1997) 311–320

Enhancements to Policy Distribution for Control Flow and Looping

Nigel Sheridan-Smith[1], Tim O'Neill[1], John Leaney[1], and Mark Hunter[2]

[1] Institute of Information and Communication Technologies,
University of Technology, Sydney
{nigelss, toneill, jrleaney}@eng.uts.edu.au
[2] Alcatel Australia
Mark.Hunter@alcatel.com.au

Abstract. Our previous work proposed a simple algorithm for the distribution and coordination of network management policies across a number of autonomous management nodes by partitioning an Abstract Syntax Tree into different branches and specifying coordination points based on data and control flow dependencies. We now extend this work to support more complex policies containing control flow logic and looping, which are part of the *PRONTO* policy language. Early simulation results demonstrate the potential performance and scalability characteristics of this framework.

1 Introduction

Policy-based Network Management (PBNM) systems require many additional capabilities in *carrier-class networks* - a service management orientation, and a capacity for *adaptive* and *dynamic* behaviour to respond to user and business needs. However, the dynamic behaviour required could cause scalability issues in large-scale networks when millions of policy decisions are required every minute for complex services. Policy-based management can potentially increase performance and scalability though the distribution of policies to multiple management nodes, but very little work has been done in this crucial area.

We are developing a policy-based service description language and management system called *PRONTO* [1] that manages dynamic and adaptive end-to-end services, whilst allowing for system evolution over time. Whilst some policies can be easily distributed according to different regions of the network, other policies will require coordination to ensure the correct sequencing of event, condition and action evaluation and information distribution. We now deal with more complex policies, involving control flow and looping. Sec. 2 briefly discusses the management system, and Sec. 3 and 4 discuss the existing work and the proposed extensions respectively. Sec. 5 presents the results of our discrete event simulator, and Sec. 6 examines related work.

2 The *PRONTO* Management System and Language

The *PRONTO* management system allows the specification of *dynamic* and *adaptive* services through a policy-based *service description language*. This

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 269–280, 2005.

language allows services to be readily changed or replaced as market requirements dictate. Each service description defines the parameters, roles (e.g. devices and systems), resources, software components, and event-condition-action (ECA) policies involved in each service. Policies are written with regard to virtual model of the devices that has hierarchically structured model elements, or software components called *domain experts* that carry out different management tasks and strategies (at different levels of abstraction and according to feedback from the network). The domain experts are interchangeable, allowing Service Providers to change management functionality to suit their individual needs.

More complex policies are required in network management to have precise control over behaviour, particularly when different customers have conditional requirements, or when different behaviours need to be assigned to different devices in the network. *PRONTO* policies supports many additional actions that are not present in other policy languages, giving the policy writer greater flexibility in describing their service and network policies. An example service is shown in Fig. 1 for the management of a full mesh of MPLS LSPs. The LSP resources are allocated when the service is put into the **enable** state, and re-requested when any given Label Switch Path fails. The 'mpls' domain expert allocates the required resources using LDP signalling. Whilst this demonstrates the versatility of the language, implementing most complex behaviour in a policy-controlled domain expert will simplify the policy specification.

```
service /mplsService {
  params {
    required CEs: collection;
    n: int = CEs.size(); }
  components {
    mpls: [/mgmt/coreMplsDomainExpert]; }
  resources { LSPs[n*(n-1)]: LSPResource; }
  events { pathFail(path): LSPs[all].failure; }

  concurrent policies {
    on (enable) {
      // Setup full mesh of LSPs between CEs
      for (int i = 0; i <n; i++) {
        for (int j = i; j <n; j++) {
          LSPs[i*n+j].signalling = LDP;
          LSPs[i*n+j].source =
            mpls.selectPhysConn(CEs.get(i));
          LSPs[i*n+j].target =
            mpls.selectPhysConn(CEs.get(j));
          mpls.consume (LSPs[i*n+j]);
    } } }

    on (pathFail) {
      // Release the existing path
      mpls.release (path);
      // Initiate another LDP path
      mpls.consume (path);
} } }
```

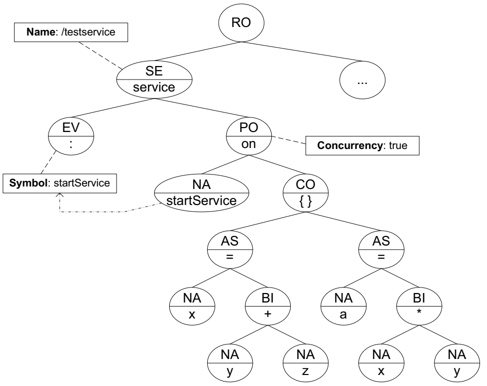
Fig. 1. Management of VPN service

3 Policy Distribution and Coordination

Some complex dynamic services, though, will put incredible demands on centralised PBNM systems. For example, the admission control function on an pay-per-channel IPTV service could be swamped with hundreds or thousands of policy decision requests during commercial breaks as users change channels repeatedly. In these cases, it is desirable to distribute such decisions to increase the

```
service /testservice
{
  events {startService;}
  concurrent policies
  {
    on (startService)
    {
      x = y + z;
      a = x * y;
    }
  }
}
```

(a) Service policies



(b) Annotated Abstract Syntax Tree (AST)

Fig. 2. Service to AST transformation

potential management load that can be handled. Our earlier work [2] described how the principles of *automatic parallelising compilers* and *Abstract Syntax Tree (AST) partitioning* could be used to distribute and coordinate policies to management nodes that are assigned the responsibility of different geographic partitions of the network. By adjusting the demarcation, the policy load on different management nodes can be adjusted. We give an overview of the basic algorithm here, but for simplicity, we only show the distribution and coordination of actions - events and conditions are treated similarly.

The **sequential** and **concurrent** keywords can be applied at different granularities to policies, to allow policy execution to be parallelised where required. However, *sequential execution semantics* are still obeyed for correct evaluation. Policies without inter-dependencies are simply distributed, as coordination is not required. Fig. 2(a) shows two policy actions, where there is a dependency between the expressions, enforcing sequential execution. We build an AST (similar to a normal compiler) as shown in Fig. 2(b): each node is marked with a type (two-letter abbreviation)[1] and the equivalent token text.

The different branches of the tree are partitioned by examining the name expression (NA) nodes that define different symbols (e.g. devices and their model elements, variable names). There are three major types of coordination messages: *data distributions* (DI) when data is exchanged, *sequence points* (SP) when sequential ordering is important, and *event notifications* (EN) to initiate distributed policy execution. We annotate the AST with additional DI and SP nodes, transforming it into a graph, so as to indicate where coordination is required between the AST partitions.

Using *depth-first traversal* to emulate sequential execution ordering of each policy action, we mark each AST node with directional arrows to indicate re-

[1] The nodes shown in this figure are as follows: Root (RO), Service (SE), Event (EV), Policy (PO), Compound Statement (CO), Assignment (AS), Name Expression (NA) and Binary Operator (BI).

gions of responsibility around different NA nodes. Descendants of NA nodes are marked with up arrows, and if all the children of any unmarked node have the same responsibility, they are marked with a down arrow. Any other unmarked nodes are assigned up arrows (or left arrows in the case of assignment operators). We then insert DI nodes between the partitions of the AST to indicate that some data needs to be exchanged between two or more management nodes, such as a calculated value or data that is retrieved from a device.

Whilst this simple process identifies some simple data dependencies, other types of dependencies go unnoticed. Wolfe [4] defines three types of dependencies: *flow dependence* when an assigned variable is later used; *anti-dependence* where a variable is used, and then reassigned; and *output dependence* when a variable is assigned twice at two different points in time. We perform a modified IN/OUT set analysis [4] to identify these dependencies; values and references are distinguished (INVAL and OUTVAL; INREF and OUTREF), and different parts of statements rather than whole statements are analysed to maximise parallelisation.

A post-order depth-first traversal is used to simulate the execution of policy expressions and statements in an sequential interpreter, and a *backpatch list* is maintained to keep track of prior IN/OUT markings, which are generated as each NA node (i.e. symbolic name) is encountered. The backpatch list stores pointers to the most recent OUTVAL for each symbol, and all INVAL/INREF markings since that NA node. Flow dependencies occur where an OUTVAL is followed by an INVAL for a symbol. A new DI node is inserted between these NA nodes to indicate this data dependency. Anti- and output dependencies occur when INVAL is followed by an OUTVAL, or OUTVAL is followed by another OUTVAL for any given non-temporary[2] symbol. New SP nodes are inserted between these NA nodes to indicate that sequencing of these actions is important, due to side-effects. Finally, setting references (i.e. OUTREF) results in one symbol *aliasing* another, and the backpatch list must maintain a pointer to the other symbol record to ensure that the history of IN/OUT markings for the correct symbol are used. Fig. 3 shows the semantic execution order and the generated IN/OUT markings and Fig. 4 shows the resulting dependency graph.

4 Extensions to the Distribution Algorithm

4.1 Control Flow Statements and Expressions

PRONTO supports statements and operators that require strict sequential evaluation order, such as **if-then-else**, **switch**, **break** and **continue** statements. It also supports exception handling, with a **throw** statement altering control flow. We continue to use sequence points, as these avoid the need for global synchronisation locking across all management nodes. Rather, sequencing only occurs between nodes that are directly inter-related.

[2] Temporary variables are implicitly duplicated, eliminating dependencies.

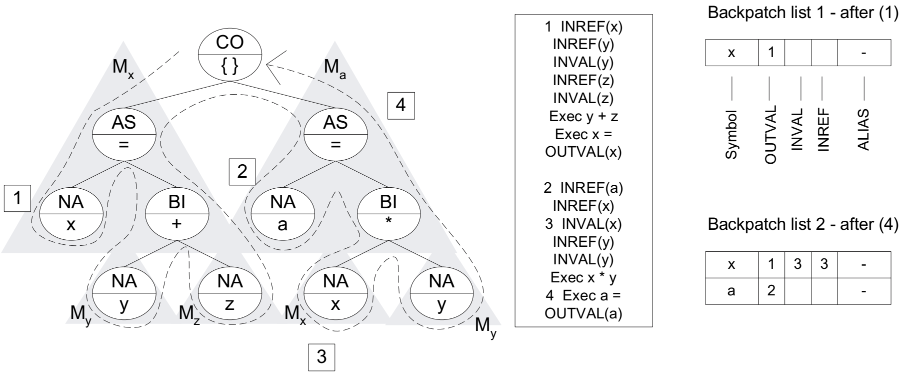


Fig. 3. Semantic execution order, IN/OUT markers and backpatching

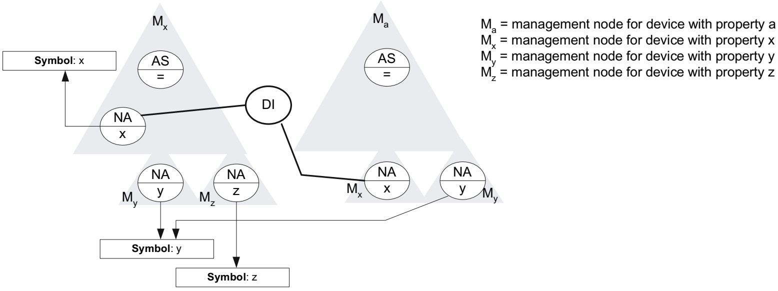
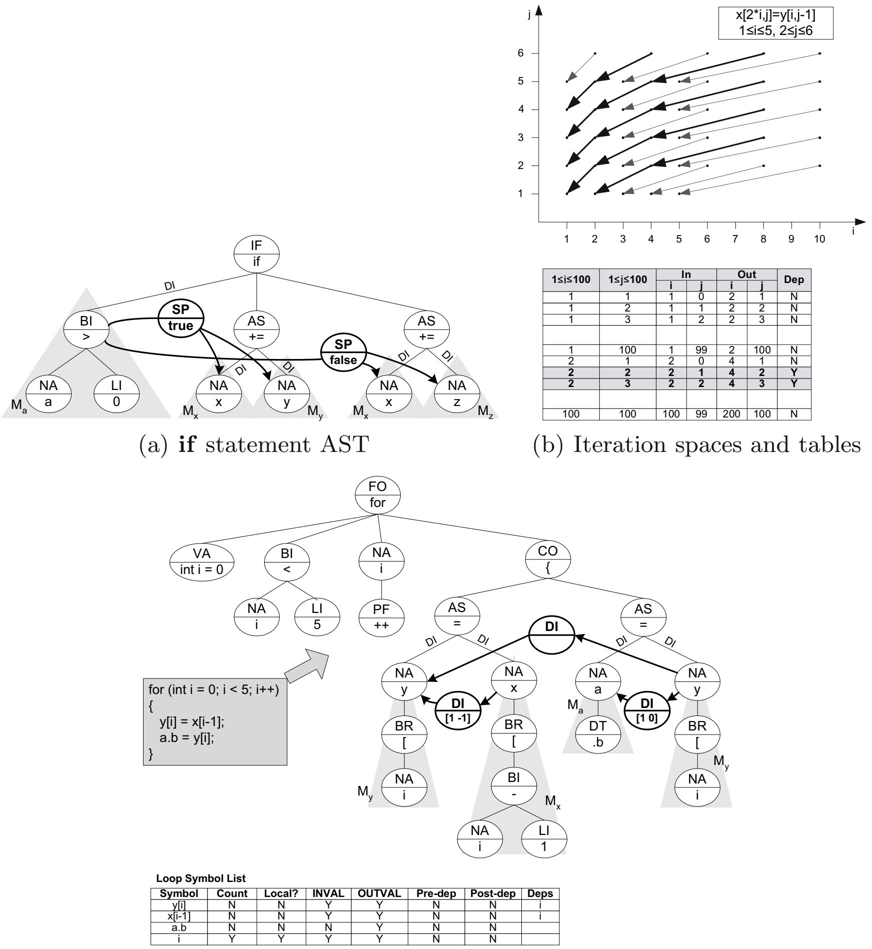

Mx
Ma
Symbol: x
AS
=
AS
=
DI
NA
x
NA
y
NA
z
NA
x
NA
y
My
Mz
Mx
My
Symbol: y
Symbol: z
Ma = management node for device with property a
Mx = management node for device with property x
My = management node for device with property y
Mz = management node for device with property z

Fig. 4. Dependency graph

In the case of **if-then-else** or **switch** statements, a SP node is inserted to connect the conditional expression to any children of the statement or expression nodes that are conditionally executed, whenever a partition boundary is crossed (i.e. at an inserted DI node). All partitions hierarchically down the tree must be associated with an appropriate SP node. Each SP node is then marked with information describing its purpose. In Fig. 5(a), the **if-then-else** (IF) node has up to three child nodes - the evaluated boolean expression, and optional true and false outcome statements. Once the conditional expression is evaluated, other management nodes are notified of the outcome. Management node M_a must notify M_x and M_y when the condition evaluates to true, or notify M_x and M_z when the condition evaluates to false. In the case of **break**, **continue** and **throw** statements, SP nodes can also be inserted into the AST to ensure that any partitions containing statements following the conditional logic are not executed, based on the outcome of the conditional expression.

4.2 Loops with Counters

Loops can often be unravelled or sub-divided to maximise parallelisation [3], depending on the presence of any *loop-carried dependencies*. The *PRONTO* lan-



(a) **if** statement AST

1≤i≤100	1≤j≤100	In		Out		Dep
		i	j	i	j	
1	1	1	0	2	1	N
1	2	1	1	2	2	N
1	3	1	2	2	3	N
1	100	1	99	2	100	N
2	1	2	0	4	1	N
2	**2**	**2**	**1**	**4**	**2**	**Y**
2	**3**	**2**	**2**	**4**	**3**	**Y**
100	100	100	99	200	100	N

(b) Iteration spaces and tables

```
for (int i = 0; i < 5; i++)
{
   y[i] = x[i-1];
   a.b = y[i];
}
```

Loop Symbol List

Symbol	Count	Local?	INVAL	OUTVAL	Pre-dep	Post-dep	Deps
y[i]	N	N	Y	Y	N	N	i
x[i-1]	N	N	Y	Y	N	N	i
a.b	N	N	N	Y	N	N	
i	Y	Y	Y	Y	N	N	

(c) **for** loop AST

Fig. 5. Complex policies and SP/DI insertion

guage has four types of loops: **for**, **foreach**, **while** and **do..while**. Different loops have various levels of complexity, so we deal with the most common varieties and revert to sequential execution when it is too difficult to predict loop behaviour. The *counter symbols* which are modified in each iteration (with OUTVAL markings) are identified, as loop-carried dependencies are reliant on the counters and are modified during loop iterations. Dependency analysis then then used to determine a suitable execution ordering of the partitions within the loop.

Firstly, we build a table of symbols contained within the loop, identifying if they are local or global symbols, if they result in INVAL or OUTVAL markings, if they have pre- or post-loop dependencies, and if they depend on other symbols (e.g. the counters). An example is shown in Fig. 5(c). Symbols updated in every iteration with no other dependencies are treated as counters[3]. We must then identify the expressions that lead to changes in the counters. If these expressions do not rely on any other symbols, then the counter behaviour can be predicted in advance, allowing localised counters at each management node.

The second stage is to identify relationships between the symbols dependent on the counters, and which loop iterations have dependencies on other iterations. To cover the majority of simple loops, we focus on array indices of the form $a_1 i + a_2 j + \ldots + a_m n + b$ when retrieving values set in previous loop iterations (i.e. INVAL operations). $i, j, \ldots, n$ are the counter variables and $a_1, \ldots, a_m, b$ are constants, and these values are stored in matrix $M = [a_1 \ \ldots \ a_m \ b]$. For example, the expression $x[2 * i - 1]$ with counters i and j has the values $a_1 = 2$, $a_2 = 0, b = -1$ and $m = 2$. Using the constructed symbol table and by examining the INVAL and OUTVAL markings on symbols (in relation to counter variable updates), we can identify flow dependencies between iterations of the loop. If the dependency is between distinct AST partitions, DI nodes are inserted into the graph between the NA nodes, indicating the direction of the flow dependency and the matrix M. The *antecedent* management node must notify the *dependent* management node of the calculated value when complete. For an example using matrices M, see Fig. 5(c).

In other cases where array indices do not fit a linear equation, or where the index equation is used on an OUTVAL operation, we can model an *iteration space* [4], simulating the loop and keeping a list of counter values that lead to dependencies. The list is then kept with the DI node added between NA nodes. The antecedent management node consults the list and notifies the dependent management node when a dependency exists - see Fig 5(b). The difficulty with this scheme is the potential table size, but non-shaded entries can be eliminated after dependencies are identified.

Special care is required when previous indexed variables are not set during the loop itself, according to sequential ordering. The expression $y[i] = y[i + 1]$ would access prior values for monotonically increasing i. We also have to ensure that prior values are retrieved before being overwritten. We can insert additional SP nodes into the AST, annotated with either the M matrix or an iteration table. Another difficulty is counters being modified in the middle of the loop - the first and last iterations are shortened, and execution of each partition must occur with the correct counter values.

4.3 Loops with Conditions

Loops based on conditional expressions require the expression to be evaluated before each iteration executes. An SP node is inserted to ensure that partitions contained within the loop are not evaluated prior to the conditional test, with notification being given on the first run and when the condition changes. This is similar to the control flow approach above. There are some difficulties though:

[3] We do not currently deal with complex counters with multiple updates per iteration or with dependencies.

how do we guarantee the loop is executed the correct number of times if there are side-effects between the conditional expression and the loop actions?

In these situations, we can enforce sequential ordering by using additional SP nodes. If the conditional expression or the loop actions are known to have side-effects that influence each other, or if it is unknown, then lock-step notifications are needed between the management nodes. This ensures that the condition is evaluated prior to other loop partitions and vice versa, so that execution occurs the correct number of times for both. Other more optimistic approaches could evaluate the conditional expression when only some of the loop actions have finished executing, or allow evaluation of the conditional expression whilst the other partitions are not lagging too far behind.

We anticipate that we might have access to additional *meta-data*, that would be beneficial in determining when side-effects might occur. Such meta-data, described as part of the management models or contained within descriptors for each vendor's devices, could indicate when property $dev001.x$ has an impact on property $dev001.y$, and would be used similarly to other meta-data for detecting and resolving conflicts. This is an area of future investigation.

4.4 Distributed Execution

Different management nodes take responsibility for different sets of devices in the network and the associated policies, and most policies are carried out autonomously. The EN, DI and SP messages must be exchanged at suitable times so that the behaviour is coordinated appropriately, depending on the policy complexity. EN messages must be sent to all participating management nodes whenever any given node recognises that a policy has been triggered, due to conditions or events that have been evaluated. The source of the EN message creates a unique execution identifier to ensure that each the policy executes exactly once, and so that later SP/DI messages apply to the same execution instance.

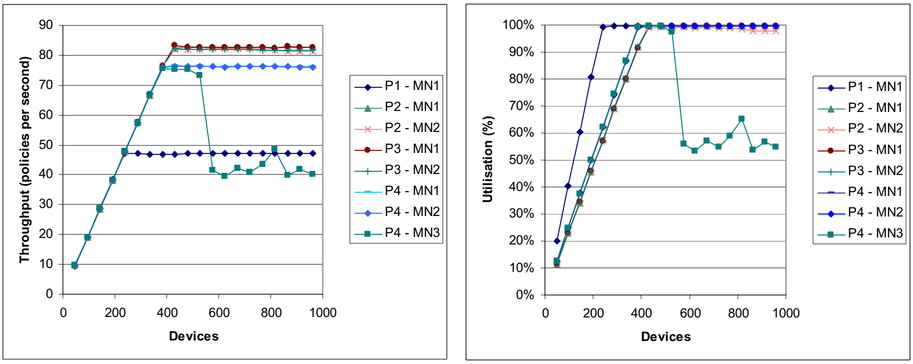
Once a management node receives a EN message, it can start executing any AST partitions that is has responsibility for, as long as they do not require an SP message (i.e. some other actions must occur first). If a DI message is required, the partition can commence execution, but the execution will halt until information that needs to be exchanged is received. Execution of the AST partition at each management node occurs independently, either sequentially or concurrently, depending on the current policy. However, localised analysis must be done, to ensure that concurrent statements are executed correctly, in the same manner as for policies that are distributed across multiple nodes. As each AST partition finishes executing, any outgoing SP and DI messages must be sent if they go to other management nodes. If the messages go to the same node, then other partitions can be triggered or finished locally. DI messages must contain calculated values which are to be exchanged between management nodes. Once each management node completes its own AST partitions, the policy finishes at each node.

5 Simulation

We are constructing a distributed policy simulator (based on OMNET++) to predict and examine the anticipated performance of this algorithm with different types of policies, with unique network and management architectures and different dynamic loads. The simulator models *execution units* representing the different partitions of the AST. Management nodes are informed of their responsibilities, and then retrieve policies and other data from a centralised relational database with ideal, localised caching at each management node. Devices generate events using an exponential distribution, resulting in policies being executed across one or more management nodes. We model the exchange of EN, DI and SP messages during execution. Each management node has a *coordinator* and *executor* with thread-pooling architectures.

Although the simulation system is not complete, we have been able to get some early simulation results based on preliminary real-system measurements and estimated model characteristics. Our initial profile consists of: device response (uniform: $\mu = 3.431ms$, $\sigma = 799.8\mu s$), database response (uniform: $\mu = 1.445ms$, $\sigma = 1.6\mu s$), cache response (exp: $\mu = 500\mu s$), coordination delay (exp: $\mu = 500\mu s$) and execution delay (exp: $\mu = 5ms$). The simulation allows the calculation of the *utilisation*, *throughput* and *response time* (measured from the initial event timestamp) at each management node.

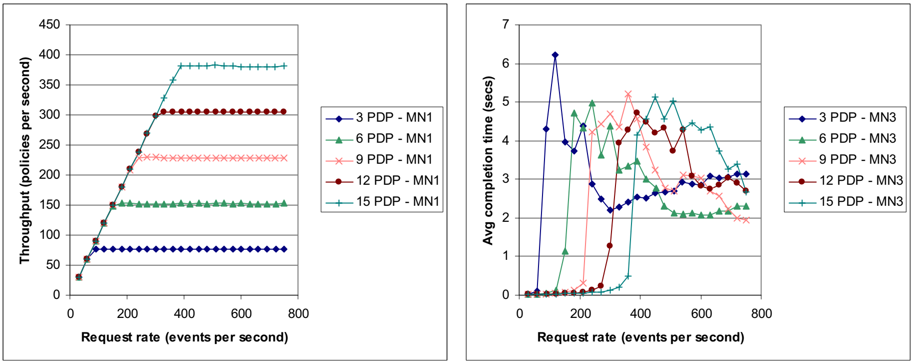
To evaluate performance, we simulate an increasing device load, with devices sending events every 5 seconds. As seen in Fig 6(a), four types of policies result in different load capacities. P1 involves setting a device property from a single node. P2 involves retrieving a device property on one node, followed by writing a property on a second node using a *sequence point*. P3 is identical to P2, except the value is exchanged between the nodes by using a *data distribution* (DI) message. P4 models retrieving two device properties on management nodes 1 and 2, followed by writing of a device property on management node 3.



(a) Throughput

(b) Utilisation

Fig. 6. Performance of single management node set



(a) Throughput

(b) Response time

Fig. 7. Scalability of management node sets

Increasing the number of management nodes increases event load capacity, but leads to a worsening of response time and capacity per node, due to coordination overheads. Furthermore, the final management node (MN3) under policy P4 is heavily overloaded by multiplexing competing events from MN1 and MN2, although selective discarding of queued messages improves the throughput and utilisation responses under excessive load conditions.

In Fig. 7(a) and Fig. 7(b), we examine scalability by simulating 540 devices with varying request rates on policy P4 (2 x reads + 1 x write over 3 management nodes), with equal device load on each management node. Here, the throughput clearly benefits from increasing the number of nodes. The response time is slightly worse with more nodes, but higher event loads are handled.

These preliminary results are promising, since they show that the distributed algorithm has the potential to scale easily, simply by adding management nodes and distributing the device load where necessary to alleviate bottlenecks. Additional nodes increases the overall throughput and load handling with a minimal impact on response time. We estimate that a single management node could handle approximately 47 policy events per second, or 240 devices with an event every 5 seconds. This compares well against the 20 events/s reported by Ponnappann et al. [5]. Eddie and Law [6] claim 400 events/s per node, although it is unclear whether caching is used and how the directory or database server is accessed.

The throughput appears to be highly sensitive to changes in the processing delays involved within each node. However, there are many factors that have an influence on performance - the complexity of the policies, the choice of language constructs used and the network and management architectures. Complex policies tend to increase the number of SP and DI coordination messages needed, and this has implications for response times. Ideally, the simulator will help in building a database of the metric cost of different policy constructs, or assist engineers in the design of dynamic and adaptive services that can meet operational load demands. Several improvements to the simulation are yet to be made, to more accurately model real conditions and more complex policies (as discussed in this paper) with different coordination strategies (e.g. *optimistic* vs. *pessimistic*).

6 Related Work

Policy distribution is occasionally mentioned in the literature but coordination has gone largely unnoticed.

The IETF policy architecture allows for some distribution through replicating PDPs and PEPs, but the responsibilities for policies must be manually configured. If two or more PDPs attempt to control the same device with overlapping configurations, the results would be unpredictable. Hamada et al. [7] attempted to increase scalability using hierarchical LDAP repositories and multicasted updates. Law and Saxena [6] also increased the scalability of the IETF architecture using transparent load-balancing agents. Corrente [8] showed that the PDP was a bottleneck in COPS-PR protocol handling. Strassner [9] introduces the concept of a *policy broker* for communication, but provides only limited suggestions for how conflict management should take place. Wies et al. [10] uses *management by delegation* to distribute policy management to extensible intelligent agents to increase scalability, but he does not characterise the global conflict resolution module proposed.

Howard et al. [11] provided separate event and policy statements, and these were deployed to separate management components by a Policy Distribution Service (PDS). The Policy Validation Service (PVS) generated events according to conditions that needed to be watched on the managed objects. However, they do not suggest how particular management components are selected. Similarly, Koch et al. [15] had separate event and policy languages, and policy objects were distributed to monitoring agents to poll managed objects and generate events when appropriate. However, a method of distributing policy actions was not discussed. Marriott and Sloman [12] distributed policy information and TCL scripts to obligation management agents using CORBA. Later work by Dulay et al. [13] allowed Ponder policies to be distributed to Policy Management Agents (PMAs) using Java RMI. However, both these two approaches require the agents to be identified by the *subject* in the policy, limiting the migration of policies between agents. Kohli and Lobo [14] created policy elements from PDL policies, which could be hosted between multiple policy servers to perform coordination. However, no clear method of doing this was specified.

7 Conclusion

We have demonstrated an algorithm for distributing policies amongst a number of management nodes based on geographical segregation, and extended this algorithm for flow control, loops and exception handling. Our early simulation results show that this approach offers great potential for scalability, with only a marginal impact on response time when management nodes are correctly provisioned for the anticipated load. The algorithm and its simulator will be immensely helpful in the design of complex, adaptive and dynamic services based on policy-based management techniques. We would like to acknowledge the generous financial support of Alcatel Australia and the Australian Research Council through Linkage Grant LP0219784.

References

1. Sheridan-Smith, N., Leaney, J., O'Neill, T., and Hunter, M.: A Policy-Driven Autonomous System for Evolutive and Adaptive Management of Complex Services and Networks. Eng. Comp. Based Sys. (ECBS 2005)
2. Sheridan-Smith, N., O'Neill, T., Leaney, J., and Hunter, M.: Distribution and Coordination of Policies for Large-scale Service Management. LANOMS (2005)
3. Grune, D., Bal, H. E., and Jacobs, C. J. H.: Modern Compiler Design. John Wiley & Sons, West Sussex (2000)
4. Wolfe, M.: High Performance Compilers for Parallel Computing. 1st edn. Addison-Wesley, Redwood City CA (1996)
5. Ponnappan, A., Yang, L., Pillai, R., and Braun, P.: A Policy Based QoS Management System for the IntServ/DiffServ Based Internet. POLICY (2002)
6. Law, K. L. E., and Saxena, A.: Scalable Design of a Policy-Based Management System and its Performance. IEEE Comm. Mag. **41**(6) (2003) 72-79
7. Hamada, T., Czezowski, P., and Chujo, T.: Policy-based Management for Enterprise and Carrier IP Networking. Fujitsu Science and Technology **36**(2) (December 2000) 128-39
8. Corrente, A., De Bernardi, M., Rinaldi, R.: Policy Provisioning Performance Evaluation using COPS-PR in a policy based network. IM (2003)
9. Strassner, S.: Policy-Based Network Management: Solutions for the Next Generation. Morgan Kaufmann. ISBN 1-55860-859-1 (2003)
10. Wies, R., Mountzia, M.-A., and Steenekamp, P.: A practical approach towards a distributed and flexible realization of policies using intelligent agents. DSOM (1997)
11. Howard, S., Lutfiyya, H., Katchabaw, M., Bauer, M.: Supporting Dynamic Policy Change Using CORBA System Management Facilities. IM (1997) 527-38
12. Marriott, D., Sloman, M.: Implementation of a management agent for Interpreting obligation policy. DSOM (1996)
13. Dulay, N., Lupu, E., Sloman, M., Damianou, N.: A Policy Deployment Model for the Ponder Language. IM (2001) 529-43
14. Kohli, M., Lobo, J.: Policy Based Management of Telecommunication Networks. Policy Workshop (1999)
15. Koch, T., Krell, C., Kramer, B.: Policy Definition Language for Automated Management of Distributed Systems. Systems Management (1996)

Author Index

Aguilar-Igartua Mónica 157
Akashi, Osamu 181
Alins, J. 157
Andrés-Colás, Jorge 145
Andrey, Laurent 24

Berrocal, Julio 1, 12
Brasileiro, Francisco 245
Burgess, Mark 97

Callejo-Rodríguez, María Ángeles 145
Cirne, Walfredo 245
Clemm, Alexander 220
Colaço, Eduardo 245

Dam, Mads 220
Danciu, Vitalian A. 84
de Albuquerque, João Porto 36
de Geus, Pauslo Lício 36
de Vergara, Jorge E. López 1, 12
dos Santos, Aldri L. 206
Doyen, Guillaume 257

Enríquez-Gabeiras, José 145

Festor, Olivier 24, 257
Figueiredo, Carlos M.S. 206
Fukuda, Kensuke 181

García-de-Blas, Gerardo 145
Guerrero, Antonio 12

Hasan 121
Hellerstein, Joseph L. 233
Hirotsu, Toshio 181
Hommel, Wolfgang 48
Hunter, Mark 269

Isenberg, Holger 36

Jornada, João 72
Julio, Berrocal 12
Jung, Gueyoung 109

Kempter, Bernhard 84
Kramer, Bill 233
Krumm, Heiko 36

Lahmadi, Abdelkader 24
Leaney, John 269
Lopes, Raquel 245
Loureiro, Antonio A.F. 206

Marques, Filipe 72
Martín, I. 157
Maruyama, Mitsuru 181
Mata-Díaz, Jorge 157
Moura, Antão 72
Mukherjee, Sarit 193

Naik, Tejas 193
Nataf, Emmanuel 257
Nogueira, José M. 206

O'Neill, Tim 269

Patterson, David 233
Pras, Aiko 60
Prieto, Alberto Gonzalez 169
Pu, Calton 109

Radziuk, Eduardo 72
Ramón-Salguero, Francisco Javier 145
Rangarajan, Sampath 193

Sahai, Akhil 109
Sampaio, Marcus 72
Sauvé, Jacques 72
Sheridan-Smith, Nigel 269
Singhal, Sharad 133
Stadler, Rolf 169, 220
Stiller, Burkhard 121
Sugawara, Toshiharu 181
Swint, Galen S. 109

Terauchi, Atsushi 181

van Wanrooij, Ward 60
Villagrá, Víctor A. 1, 12

Wang, Zhikui 133
Wu, Qinyi 109
Wuhib, Fetahi 220

Xu, Wei 233

Yan, Wenchang 109

Zhu, Xiaoyun 133