LNCS 4195

Dominique Gaïti Guy Pujolle
Ehab Al-Shaer Ken Calvert
Simon Dobson Guy Leduc
Olli Martikainen (Eds.)

Autonomic Networking

First International IFIP TC6 Conference, AN 2006
Paris, France, September 2006
Proceedings

ifip

Springer

Dominique Gaïti Guy Pujolle
Ehab Al-Shaer Ken Calvert
Simon Dobson Guy Leduc
Olli Martikainen (Eds.)

Autonomic Networking

First International IFIP TC6 Conference, AN 2006
Paris, France, September 27-29, 2006
Proceedings

Springer

Volume Editors

Dominique Gaïti
Université de Technologie de Troyes,Institut Charles Delaunay, France
E-mail: Dominique.Gaiti@utt.fr

Guy Pujolle
University of Paris 6, LIP6 Laboratory, Paris, France
E-mail: guy.pujolle@lip6.fr

Ehab Al-Shaer
DePaul University, http://www.depaul.edu, USA
E-mail: ehab@cs.depaul.edu

Ken Calvert
University of Kentucky, http://www.uky.edu, USA
E-mail: calvert@netlab.uky.edu

Simon Dobson
Systems Research Group, UCD Dublin, Ireland
E-mail: simon.dobson@ucd.ie

Guy Leduc
University of Liège, Electrical Engineering and Computer Science, Belgium
E-mail: Guy.Leduc@alg.ac.be

Olli Martikainen
University of Oulu, Finland
E-mail: olli.e.martikainen@kolumbus.fi

Library of Congress Control Number: 2006932797

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

LNCS Sublibrary: SL 5 – Computer Communication Networks
and Telecommunications

ISSN 0302-9743
ISBN-10 3-540-45891-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-45891-3 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11880905 06/3142 5 4 3 2 1 0

Preface

The autonomic communication paradigm has been defined mainly through the Autonomic Communications Forum (ACF) and particularly as follows: Autonomic communication is centered on selfware – an innovative approach to perform known and emerging tasks of a network control plane, both end-to-end and middle box communication-based. Selfware assures the capacity to evolve; however, it requires generic network instrumentation. Selfware principles and technologies borrow largely from well-established research on distributed systems, fault tolerance among others, from emerging research on non-conventional networking (multihop ad hoc, sensor, peer-to-peer, group communication, etc.), and from similar initiatives, such as Autonomic Computing of IBM, Cognitive Network of DARPA, Harmonious Computing of Hitachi, Resonant Networking of NTT, etc.

A visionary network would be able to (a) configure and re-configure itself, (b) identify its operational state and take actions to drive itself to a desired stable state and finally (c) organize the allocation and distribution of its resources. To build such a network, it is necessary to go beyond the improvement of techniques and algorithms by using a new concept, the knowledge plane. The knowledge plane is able to collect information available in the network to provide other elements of the network with services and advice and make the network perform what it is supposed to. There are many objectives to the configuration and reconfiguration of the network, from the optimization of resources to the use of best available techniques in order to offer the most appropriate service, best adapted to the terminal capabilities.

The goal of Autonomic Networking 2006 was to bring together researchers and practitioners to discuss the latest developments in this area of autonomic communications applied to networking. This development spans both the theoretical and the practical aspects of the domain.

The Autonomic Networking Conference is the first international conference on all the aspects of autonomic networking (architecture, services, tools, security, communications, etc.). This conference groups four past events: SMARTNET focused on tools for autonomy, INTELLCOMM on autonomic management and services, IWAN on active networks and WAC on autonomic communications.

We hope you will enjoy the papers and find this book a valuable addition to your library.

Dominique Gaiti
Guy Pujolle
Ehab Al-Shaer
Ken Calvert
Simon Dobson
Guy Leduc
Olli Martikainen

Table of Contents

AN'06

Autonomic Networks

Self-configuration

Autonomic Platform and Services

Autonomic Management and Discovery

Policy-Based Management

Ad Hoc, Sensor and Ambient Autonomic Networks

Autonomic Control of Mobile Networks

Towards Autonomic Networks

S. Schmid[1], M. Sifalakis[2], and D. Hutchison[2]

[1] NEC Europe Ltd., Network Laboratories, 69115 Heidelberg, Germany
schmid@netlab.netlab.nec.de
[2] Computing Department, Lancaster University, Lancaster, LA1 4WA, U.K.
{mjs, dh}@comp.lancs.ac.uk

Abstract. Autonomic networking set a challenge for the research community to engineer systems and architectures that will increase the QoS and robustness of future network architectures. However, our experience is that so far the autonomic network research community does not have a common perception of what an *autonomic network* is. This paper attempts to propose a generic model for *autonomic systems*, along with a minimum set of required properties that would render a system compliant to this model. The paper emphasises the importance of such a common model for the credibility of the research community as well as to eliminate attempts to unnecessarily overload or blur the scope of the field.

Keywords: Autonomic communication, autonomic networks, autonomic system definition.

1 Introduction

Over the last 15 years, the on-going convergence of networked infrastructures and services has changed the traditional view of the network from the simple wired interconnection of few manually administered homogeneous nodes, to a complex infrastructure encompassing a multitude of different technologies (wired/wireless, mobile/fixed, static/ad-hoc, etc.), heterogeneous nodes (regarding size, capabilities, power and resources constraints, etc.), diverse services (end-to-end, real-time, QoS, etc.), and competing objectives which are subject to "tussles-spaces" of interests [CWSB02]. This situation has put a challenge for the research community to engineer systems and architectures that will increase the QoS and robustness of the current and future internetwork whilst alleviating the management cost and operational complexity.

The autonomic communications research community [ACF] has been formed (among others) to respond to this challenge. Based on interdisciplinary grounds, it tries to tackle the problem by developing architectures and models of (networked) systems that can manage themselves in a reliable way always fulfilling their service mission. The need for such genuine course of action is also reflected in the memorandums of the initiatives that fund this research, in the EU [IST-FET] and world-wide (e.g. [NSF-FIND]).

Given the interdisciplinary grounds of the research area, it is very hard to argue about the originality of the practices, methods and theories that are anticipated by this

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 1 – 11, 2006.

work. There are a number of areas (see section 2) where their investigators would argue falls into the field of autonomic research. However, this paper emphasises that what characterises an area as novel/pioneering is the originality of its goals and objectives more than the means and methodologies used to achieve them. Besides, it is not surprising for different research communities to make use of the same or similar "tools" in order to achieve better solutions in their field. As a result, in order to improve the credibility of the autonomic research community as well as to eliminate attempts to unnecessarily overload or blur the scope of the field, this paper emphases the importance to have a clear and common view of what the research goals are and how to validate that. As a result, it is imperative that the research community agrees on what "autonomic" is and what it is not, and to have a way of accessing this functionality in a system.

Based on extensive discussion of this topic at various academic events [Dag06011][Infocom06][IWAN05][WAC05] and within the IST funded Autonomic Network Architecture (ANA) project [IST-ANA], this paper attempts to capture a definition for an autonomic system by proposing a model and identifying its essential properties, and advocates the need for a "test" that will enable the assessment of autonomic behaviour in a system. It is important to stress here that this paper does not claim to provide a complete definition, but it should rather be considered as a starting point for identifying the scope and the objectives of the research area.

Section 2 provides a short overview of early work in relevant areas, for example, autonomic computing. Section 3 provides some linguistic grounds to support the definitions that are propose in section 4. There it is also discussed what the essential properties of an autonomic system are. Section 5 advocates the need for a test to assess autonomicity in a system. Finally, section 6, concludes the papery with a summary of the work.

2 Background

Much of the research that has taken place in the past in various fields of computer science and network research could potentially classify as autonomic systems research (as argued by its investigators). Yet, as no standard definition exists to formally describe an autonomic system and its expected properties, the usage of the term remains vague and there is not conclusive way of assessing to what extend a research area qualifies as autonomic systems research and when the research objectives have been met. Nevertheless, there are several research areas that are relevant to the work on autonomic systems in a more or less direct way. This section points out a few of them.

Obviously and most probably the most relevant research work to autonomic systems seems to be the *Autonomic Computing* initiative of IBM through the e-Lisa project. Driven by their business market needs in 2001, IBM announced their intention to develop systems that are "autonomic". Inspired by the operation of the autonomic nervous system (ANS) in biological systems, they specified four properties that describe autonomicity in a system. These are:

- *Self-healing*: discover and repair potential problems to ensure that the system runs smoothly.
- *Self-protection*: identify threats quickly and take protective actions. Sensors feed data to a protection centre, for auditing, and action taking against various threats.
- *Self-configuration*: install and set up applications/patches/updates automatically, verify compliance with the specified service levels, optimise configuration of applications using adaptive algorithms.
- *Self-optimisation*: constantly monitor predefined system goals and performance levels to ensure that all systems are running at optimum levels.

In spite of the kudos created around this initiative and the seemingly visionary research effort by the sounds and the theory of it, IBM's vision proved to be rather short-sighted and focused on delivering simply software applications and application frameworks that could auto-configure, download updates, and occasionally learn user preferences or adapt to usage patterns with minimal administrative intervention, as opposed to infrastructures or architectures (software, computer systems, networks, etc) that can be self-managed and self-sustainable at software (system and application) and hardware level. Lest the term "computing" instead of "systems" in the definition.

Similar to IBM's effort to deliver "autonomic" business solutions are other projects by other market competitors such as Adaptive Enterprise [HPAE] by HP, Dynamic Systems Initiative [MSDSI] by Microsoft, eBusiness on Demand [IBMBOD] by IBM, N1 [SUNN1] by Sun Microsystems, Organic IT [FROIT] Forrester Research.

This paper differentiates between *autonomic systems* and IBM's *autonomic computing*. The reason being twofold: first, the term autonomic computing as defined by the IBM does not capture the complete spectrum of properties and behaviour an autonomic system should exhibit, and second by limiting our analysis to "systems" (whether that is a host, a network, or a complete infrastructure), the focus can be on well defined abstractions that are analysed, characterised and finally specified (section 4). This is in contrast to the term "computing" which has fuzzy and broad semantics and can include more or less anything and everything ranging from algorithms and software engineering methods to software and hardware systems.

Other areas related to autonomic systems include active and programmable networks [TW96, TSSM97]. The main connecting element between the two areas is the perceived need for adaptation and customisation of a system. An autonomic system must be able to customise the self in response to changes in policies or mission or in order to adapt to the deployment environment. Active and programmable platforms (e.g. [SFSS01, KRGP02, Hja00]) provide a means for generic adaptation and customisation. However active or programmable systems alone are not to be considered autonomic. Similarly to programmable systems, composable architectures (e.g. [HP91, MKJK99]) and component models (e.g. [CBGJLU04]) provide a degree of adaptiveness and customisability that is absolutely essential to autonomic systems.

Many people in the research community influenced by the exhibited autonomicity in biological systems tend to believe that an autonomic system needs to have intelligence and cognition. To that extend research on artificial intelligence and robotics

[Bro91] maybe closely related to autonomic systems. After all one of the long haunting visions of AI, has been to deliver robotic systems that think and act for themselves in the environment, i.e. autonomic systems per se. Evolutionary strategies [Sch01, Koz92] and probabilistic learning [Bis95] may offer sustainable solutions for adaptation in operational conditions that could not be determined, evaluated or prescribed at system design time.

While the authors fully support the argument that intelligence, learning and cognition can augment autonomic systems, this paper examines to what extend intelligence is an axiomatic property for autonomicity. On one hand experience has shown that deterministic systems and analytic methods may perform poorly in finding solutions for problems or conditions outside their initial design space therefore necessitating a degree of evolvability that draws from intelligence. On the other hand lessons learned from control theory show that it is possible to develop highly adaptive systems to unthought operational conditions without need for architectural evolvability. The main question to ask here is if one of the axiomatic properties of an autonomic system is the ability to evolve its architecture or simply adapt its operation in face of conditions that extend its original design space.

Finally another research area which often considered relevant to autonomic systems involves dependable and resilient systems (e.g. Grids). This is due to the inherent need of such infrastructures for persistence and survivability with low management effort and often in an unsupervised manner. An interesting point to notice here is that very often in these systems the building elements of are not autonomic individually but the whole system might be behaving autonomously collectively (from a high level management perspective). A misconception, which in our opinion is the differentiating point between the two research areas is the belief that an autonomic system is one which is self managed and “always ON”, tolerant to attacks, problems, and condition changes. While it is true that an autonomic system must exhibit a reactive behaviour to counter problems, attacks and operational condition changes, it is not very clear what this reaction will entail. In some occasions it may be sensible for an autonomic system to hibernate during an “operational winter” period (e.g. after a natural disaster where the power may be a scarce resource) or even self-destroy depending on the mission (e.g. to avoid having secret codes intercepted or being hijacked). This behaviour might be contradicting with the principle of resilience or “always ON” operation. Although solutions are possible in all these cases our aim is to point out the potentially conflicting objectives of the two research areas which should not be identified as overlapping.

3 Some Etymology

A common misunderstanding comes from the fact that researchers believe autonomicity is the same as robustness, adaptability, intelligence and dependability, and as a result argue that research into autonomic systems is nothing new. However, this paper highlights that autonomicity, although related to the above topics, is something semantically different and new if considered holistically. The paper therefore tries to build clear understanding of what “autonomic” means and what the expected properties of an autonomic system are.

From a linguistic point of view the following definitions are relevant:

- *Automatic* [Greek automatos: auto- (*auto)* + -matos (*willing)*] refers to an act happening or an entity existing through the operation of a pre-existing arrangement (law of enforcement) that is triggered by some event.
- *Autonomous* [Greek autonomos : auto- (*auto)* + nomos (*law)*] means self-governed and refers to an entity capable of responding, reacting, or possibly developing independently of the whole, in mind or judgment and not controlled by others or by outside forces
- *Adaptive* means to posses the ability to change and make suitable to or fit for a specific use or condition.
- *Aware* means to have knowledge or experience of something and so being well informed of what is happening in that subject (state) at the present time. In accordance to this definition *self-awareness* means to have good knowledge and judgment about yourself

Although *autonomic* has the same roots as the word *autonomous* and is often used interchangeably (and consequently so advocated in IBM's terminology), from another linguistics point of view, autonomic lies between "autonomous" and the Greek word "autonomistic", which in turn refers to the tendency to be autonomous under any circumstance, functionally independent and under no voluntary control.

4 Defining Autonomicity

Although research in autonomic communications has its focus on network and communications aspects, the discussion regarding a definition for autonomicity aims to be more general and therefore tackles the problem from an abstract viewpoint. In this way, our definition tries to capture the semantics of autonomic systems in general, independent from whether it is software or hardware, an end system or a network system, etc.

From an architectural point of view, a *system* has a set of *inputs* and a set of *outputs*. A system implements some type of processing function, and based on its inputs, it generates some output. Zero inputs give the system semantics of a *source* and zero output semantics of a *sink.*

The following definition of an *autonomic system* stems from:

a) the semantics conveyed in the etymological study presented in section 2,
b) the empirical understanding of equivalent biological systems (ANS, eco-systems, living beings), and
c) the perception of the "common usage" of the term by the research community (i.e. considering those characteristics that are commonly associated with the term).

Proposed Definition

An *Autonomic System* is a system that operates and serves its purpose by managing its own self without external intervention even in case of environmental changes. A conceptual model of an autonomic system is shown in (Figure 1).

A fundamental building block of an autonomic system is the sensing capability (*Sensors Si),* which enables the system to observe its external operational context. Inherent to an autonomic system is the knowledge of the *Purpose* (intention[1]) and the *Know-how* to operate itself (e.g. boot-strapping, configuration knowledge, interpretation of sensory data, etc.) without external intervention. The actual operation of the autonomic system is dictated by the *Logic,* which is responsible for making the right decisions to serve its *Purpose*, and influence by the observation of the operational context (based on the sensor input).

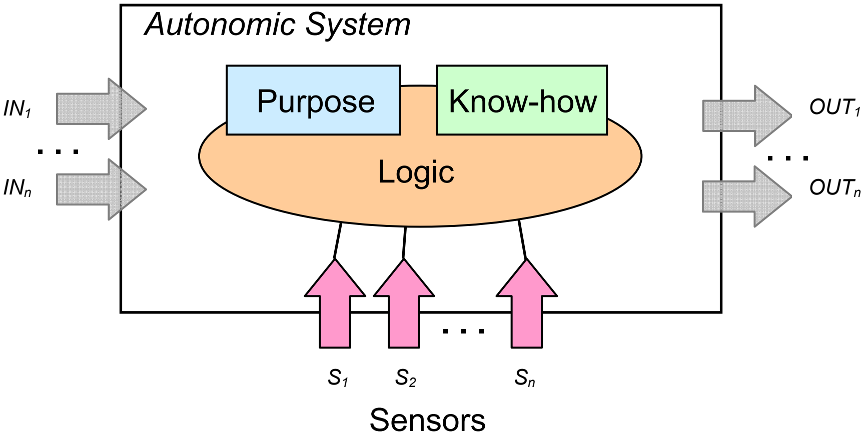


Fig. 1. Conceptual Autonomic System Model

This definition highlights the fact that the operation of an autonomic system is purpose-driven. This includes its mission (e.g. the service it is supposed to offer), the policies (e.g. that define the basic behaviour), and the "survival instinct". If seen as a control system this would be encoded as a feedback error function or in a heuristically assisted system as an algorithm combined with set of heuristics bounding its operational space.

Even though the purpose and thus the behaviour of autonomic systems vary from system to system, every autonomic system should be able to exhibit a minimum set of properties to achieve its purpose:

- **Automatic**
  This essentially means for a system to be able to self-control its internal functions and operations. To be truly automatic (i.e. even to start-up the system) implies that a system is self-contained and able to bootstrap and operate without any manual intervention or external help (i.e. the *Know-how* required to bootstrap and operate the system must be inherent to the system).
- **Adaptive**
  An autonomic system must be able to change its operation (i.e. its configuration, state and functions). This will allow the system to cope with temporal &

[1] Note that *intention* refers to the purpose or aim of an autonomic system.

spatial changes in its operational context either long term (environment customisation/optimisation) or short term (exceptional conditions such as malicious attacks, faults, etc.).

- **Aware**
  An autonomic system must be able to monitor (sense) its operational context as well as its internal state in order to be able to assess if its current operation serves its purpose. Awareness will control adaptation of its operational behaviour in response to context or state changes.

Notice that the above definitions are still in-line with IBM's basic vision of *autonomic computing*. However, in contrast to the herein proposed definition, which defines the properties that characterise an autonomic system, IBM's four "autonomic *self*-properties", namely self-healing, self-configuration, self-optimisation and self-protection, define a set of functionalities or features that an autonomic system must provide. For example, according to our definition a system that can operate on its own while serving its purpose is autonomic, irrespective of whether it implements those functionalities. In our view, it should be the nature and purpose of an autonomic system that defines which functions are required! As a result, it can be argued that the herein proposed definition is more precise (in describing a system) and at the same time more general.

The above model is also consistent with the definition of a control system as well that of an AI agent. This is reasonable since practically a control system or an AI agent may easily implement an autonomic system, if it exhibits the aforementioned properties.

Finally, a short remark regarding the relation of autonomicity and evolvability is provided, since it has been often argued in discussions that an autonomic system ought to be evolvable (for example, through some type of artificial learning methods). Similarly to the above discussion regarding IBM's autonomic computing features, it can be argued that learning and evolvability may be a useful feature in an autonomic system, but whether it is required or not depends on the actual purpose of the autonomic system, and hence should not be considered an essential property of an autonomic system.

4.1 Why Are These Properties Essential?

The question whether the three properties (automatic, adaptive, aware) are essential for autonomic systems or not requires some further discussion. This section follows a *reduction ad absurdum* approach to show that all three of them are essential**.**

First of all let's consider a system that is not automatic, yet adaptive and aware. Such a system has knowledge of its operational context and internal state, and can adapt current operation in response to context changes. However it cannot bootstrap on its own, or initiate functions that it has not used (needed) before, or terminate functions that it does not need anymore without external intervention. As a result, sooner or later it will not be able to serve its purpose anymore, either because it cannot start, or because it will not be able to perform a required function that it has not activated before (e.g. to counter an attack), or because it may run out of resources. Therefore such a system cannot qualify as autonomic.

If a system is automatic and aware but not adaptive, then it can bootstrap alright and control its possible operations, as well as sense the environment and know its internal state. However, if the environment changes, although the change will be detectable, the system will not be able to decide what steps it needs to take in order to adjust accordingly its operation with regard to serving its purpose. The ability to sense the environment and monitor internal operation (awareness) and to control the operation (automatic) is not necessarily sufficient for the system to be able to service its purpose. Operational changes involving internal (re-)configuration and/or state changes will not be feasible if a system lacks the ability to adapt.

Finally let's assume a system which is automatic and adaptive but not aware. Following a similar line of argumentation it is easy to infer that such a system will not be autonomic because it will not be able to trigger adaptation and eventually not survive in face of environment changes. Most important, such a system has no way of sensing and assessing if its operation complies with its *purpose*.

As a result of this analysis, it can be concluded that all three properties are essential for a system to be autonomic.

4.2 Autonomicity and Deterministic Behaviour

The question whether an autonomic system needs to behave/operate in a deterministic way has often been raised. For an autonomic system to be deterministic would imply that its behaviour will be fully predictable, and thus its compliance with its purpose could be verified.

According to our definition, deterministic behaviour is not one of the essential properties of an autonomic system. The reason lies in the conjecture that in any non-analytical or non-combinatorial system, although it is not possible to determine or predict the "next state change" or "action", the system can still be engineered in such a way that it will comply with its purpose (using bounds or heuristics). This is also true for biological systems, where often in similar situations different decisions are made, but still to achieve the same objective (e.g. hunt to feed).

Deterministic behaviour and autonomicity should therefore be considered orthogonal concepts. Whether or not an autonomic system is deterministic depends on the logic that defines the system behaviour, which in turn depends on the type of system and its purpose. For example, if the logic of an autonomic system is able to self-evolve through learning, the behaviour of the system easily becomes non-deterministic. On the contrary, if the logic is defined by a fixed finite state machine, the systems behaviour is deterministic and therefore traceable.

In general it may be desirable to have a deterministic autonomic system, as one can easier verify its operation and behaviour. On the other hand, this may also limit an autonomic system in a way. For example, it doesn't allow the system to learn from its "experiences" and self-evolve.

5 Testing Autonomicity

Up to now, a model for understanding the basic functionality and properties that are required by a system to exhibit an autonomic behaviour has been proposed. However, when implementing an autonomic system one needs to be able to verify and assess to what extend the goal is met and if the modelled functionality is reflected in the operation of the system.

This is a particularly difficult problem to address, which can be partly attributed to the lack of empirical experience with real platforms or simulation models of autonomic systems due to the young life of the research field.

One possible approach to address this problem is by testing one by one the four properties that characterise an autonomic system. Separate simple tests could be developed that could be used to test the behaviour of a system in different scenarios and verify to what extended the system is automatic, adaptable and aware. Then a more synthetic test should ascertain the soundness of the expected interactions (involving the *logic*) between these four properties in fulfilling the *purpose* of the system.

Another approach would be to seek for a more abstract "Turing-like" test whereby the system is seen as a black box, with the only verifiable being what one can perceive externally: (a) *is the system is still operational?* (b) *is the expected service is still provisioned?* However, like in the original Turing test there is no possible way of testing it against all possible cases.

A last approach is to continuously test the system at run-time (on-line test). A test function can be periodically applied in the system to assess its compliance with the *purpose*. In this case the challenge is to determine a set of metrics to "measure" automatically the behaviour of the system. These metrics would need to be developed on a case per case (system) basis.

Testing evolvable and thus potentially non-deterministic autonomic systems is even more challenging, since in this case it is not possible to simulate the *logic* and test the system off-line (only on-line tests may be applicable). Practically, there is no obvious way of validating the correctness of the autonomic behaviour of a self-evolved system (as it is unclear how the systems changes its operational pattern), and therefore it may not even be possible to identify a self-evolving autonomic system.

6 Conclusions

The main objective of this paper is to identify and demonstrate the need for means of accessing to what extend a system exhibits an autonomic behaviour. First, the paper proposes and analyses a model for an *autonomic system*. Then, a set of essential properties that render a system compliant to this model have been identified. Finally, a set of possible options for testing these properties in order to inspect the autonomicity of a system have been discussed. The incentive for this lies in the argument that unless there is a common understanding of what is expected from an autonomic system and how to assess autonomicity, there is no common ground to drive research in the area of autonomic communication and networking. Such a foundation is also needed to establish a credible community with well defined objectives, tools and methodologies. This is of fundamental importance to sustain longer term research and evaluate its success.

Acknowledgments. This work has originated within the ANA project which is an EU-IST funded project under the FET initiative. We would also like to thank many of the participants of the Schloss Dagstuhl seminar 06011 on *"Autonomic Networks"* for inspiring and motivating this work.

References

[ACF] Autonomic Communications Forum. Online at: http://www.autonomic-communication.org/

[Bis95] C. M. Bishop. “Neural Networks for Pattern Recognition”. Oxford University Press (1995).

[Bro91] Brooks, R.A. "How to build complete creatures rather than isolated cognitive simulators," in K. VanLehn (ed.), Architectures for Intelligence, pp. 225-239, Lawrence Erlbaum Assosiates, Hillsdale, NJ, 1991.

[CBGJLU04] Coulson, G., Blair, G.S., Grace, P., Joolia, A., Lee, K., Ueyama, J., “A Component Model for Building Systems Software”, In *Proc. IASTED Software Engineering and Applications* (SEA’04), Cambridge, MA, USA, Nov 2004.

[CWSB02] Clark, D. D., Wroclawski, J., Sollins, K. R., and Braden, R. 2002. Tussle in cyberspace: defining tomorrow's internet. In *Proc. of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (SIGCOMM 2002), Pittsburgh, Pennsylvania, USA, 2002.

[Dag06011] International Conference and Research Center for Computer Science, Schloss Dagstuhl, Wadern, Saarland, Germany, January 2006.

[FROIT] Organic IT. Forrester Research. Online available at: http://www.forrester.com/ Research/Document/ Excerpt/ 0,7211,34342,00.html

[Hja00] Hjalmtysson, The Pronto platform - a flexible toolkit for programming networks using a commodity operating system. In Proc. of the International Conference on Open Architectures and Network Programming (OPENARCH 2000), March 2000.

[HP91] N. C. Hutchinson and L. L. Peterson, "The x-kernel: An Architecture for Implementing Network Protocols," IEEETransactions on Software Engineering, vol. 17, January 1991.

[HPAE] Adaptive Enterprise Services and Adaptive Network Architecture. HP Invent. Online available at: http://h20219.www2.hp.com/services/cache/77602-0-0-0-121. html

[IBMBOD] E-Business On Demand. IBM Research. Online available at: http://www-306.ibm.com/e-business/ ondemand/us/index.html

[Infocom06] 25th Conference on Computer Communications (IEEE Infocom), Barcelona, Spain, April 2006.

[IST-FET] Future and Emerging Technologies Program. Information Society Technologies. Online available at: http://cordis.europa.eu/ist/fet/home.html

[IST-ANA] AutonomicNetwork Architecture Project. Online available at: http://www.ana-project.org

[IWAN05] 7th International Working Conference on Active Networks (IFIP IWAN), Sofia Antipollis, Cote d' Azour, France, November 2005.

[Koz92] Koza, J.R., Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press 1992.

[KRGP02] Keller, R., Ruf, L., Guindehi, A., Plattner, B., PromethOS: A Dynamically Extensible Router Architecture Supporting Explicit Routing. In Proc. of the 4th International Conference on Active Networks (IWAN), Zurich, Switzerland, December 4-6, 2002.

[MKJK99] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The Click modular router. In Proc. of the 17th ACM Symposium on Operating Systems Principles (SOSP '99), pages 217--231, Kiawah Island, South Carolina, December 1999.

[MSDSI] Dynamic Systems Initiative. Microsoft Corporation. http://www.microsoft.com/windowsserversystem/dsi/default.mspx

[NSF-FIND] Future Internet Network Design Initiative. National Science Foundation. Online available at: http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=12765

[Sch01] Schmitt, Lothar M (2001), Theory of Genetic Algorithms. Theoretical Computer Science (259), pp. 1-61.

[SUNN1] N Computers Operating as 1. Sun Microsystems. Online available at: http://www.sun.com/software/ learnabout/n1/

[SFSS01] Schmid, S., Finney, J., Scott, A., C., Shepherd, W., D., Component-based Active Network Architecture, In Proc. of IEEE Symposium on Computers and Communications, July 2001.

[TW96] D. L. Tennenhouse and D. J. Wetherall. Towards an active network architecture. In Proc. of Multimedia Computing and Networking '96, January 1996.

[TSSM97] DL. Tennenhouse, JM. Smith, W. D Sincoskie, DJ. Wetherall, and G J. Minden. A Survey of Active Network Research. *IEEE Communications Magazine*, Vol. 35, No. 1, pp80-86. January 1997.

[WAC05] 2nd International Workshop on Autonomic Communication (WAC IFIP 2005). Vouliagmeni, Athens, Greece Oct. 2005.

A Cognitive Architecture for Personal Networks

Yunfei Wu and Ignas Niemegeers

WMC group, Department of Electrical Engineering,
Delft University of Technology,
Mekelweg 4, 2628 CD Delft, The Netherlands
{y.wu, I.G.M.M.Niemegeers}@ewi.tudelft.nl

Abstract. This paper proposes a three-layer cognitive architecture for pervasive and intelligent computing of personal networks. The key element of the proposed architecture is the cognitive layer, which consists of five components, namely the context cognition, the personalization cognition, the resource cognition, the network cognition, and the cognition management. In order to demonstrate the purpose of each one of the cognition component, we present a motivation example on session mobility and show that the proposed architecture enables proactive configuration and thus hides the latency of configuration. Finally, this paper identifies the research issues that need to be addressed in order to implement the cognitive architecture for personal networks.

1 Introduction

A Personal Network (PN) [1] is a distributed personal environment where people interact with various companion, embedded, or invisible computers not only in their close vicinity but potentially anywhere. By creating a personal distributed environment, a PN extends and complements the concept of ubiquitous computing with a user-centered view. Figure 1 illustrates the concept.

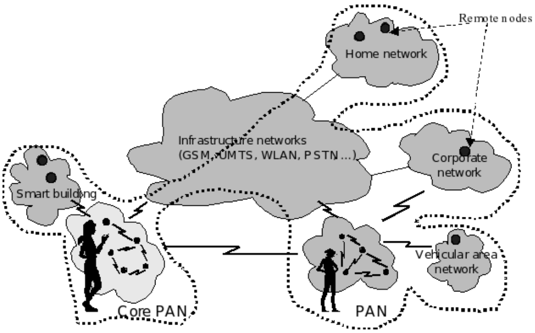


Fig. 1. An example of a Personal Network

Mark Weiser, the originator of the concept of ubiquitous computing, has envisioned that computing technologies would disappear into the background and weave

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 12 – 24, 2006.

themselves into the fabric of everyday life [2]. In the case of a distributed personal environment, such disappearance requires context awareness and the ability of self-organization. Personal environments need to be smart to acquire and apply knowledge about a person and his surroundings in order to learn his preferences, to be proactive under uncertainty, and to be secure from malicious attacks. This requires some intelligent behavior in the personal environment.

As a consequence, a new research topic, the cognitive networking, is emerging, exploiting machine learning in networking. An example is the knowledge plane [3], a concept aiming to utilize cognitive techniques (e.g., representation, learning, and reasoning) to make the network aware of what is going on in the network and respond. A PN, however, not only needs to be smart and intelligent in managing itself, but also in providing personal services. Therefore, it must be capable of learning the preferences of its owner, reasoning about what he intends to do, and acting proactively. The goal of this paper is to explore how cognitive technology could be applied in a PN for providing personal services and managing itself. To accomplish this, we propose a high level abstraction of the three-layer cognitive architecture, namely *the device layer, the cognitive layer, and the service layer*. The key new element of the proposed architecture is the cognitive layer. It consists of five components: *context cognition, personalization cognition, network cognition, resource cognition, and cognition management*. The proposed cognitive architecture provides the functionality to manage intelligent components, enable communication between them, and facilitate the interfaces between a person and his environment.

This paper is organized as follows. Section 2 presents some related work to this paper. Section 3 describes the main characteristics of PNs. Section 4 presents the main capabilities the cognitive architecture should provide in a PN environment. Section 5 proposes a cognitive architecture, stressing the functionalities to be provided and the interfaces among the different components. Section 6 illustrates the role of the architectural components. Section 7 discusses open questions regarding the implementation of the proposed cognitive architecture. In Section 8, we draw some conclusions.

2 Related Work

There are a number of research being worked on cognitive networking. IBM [4] has proposed autonomic computing systems: they manage themselves and free system administrators from many of today's routine management tasks. Self-management in their view includes four aspects: self-configuration, self-healing, self-optimization, and self-protection. Dietterich et al [5] examine various aspects of machine learning that are useful for a cognitive approach to networking and they review three capabilities in cognitive networking. These are anomaly detection and fault diagnosis, responding to intruders and worms, and rapid configuration of networks. Works have been going in various projects to design pervasive computing systems and applications that adapt themselves to their users, e.g., Project Aura at Carnegie Mellon University [6], Easyliving project [7] at Microsoft, and MIT Project Oxygen [8]. However, our work aims at designing an intelligent environment specifically for PNs with the knowledge of the characteristics of PNs.

3 The Main Characteristics of Personal Networks

Before we describe the cognitive architecture design, let us present those characteristics of a PN that have a strong influence on its design.

- **Heterogeneity:** Heterogeneity exists in a PN in a number of ways, such as the variety of devices, the geographical distribution of its components, the diversity of wireless and wired interfaces, and the different resource constraints. A PN consists of heterogeneous devices including diverse sensors and actuators, digital cameras, mobiles, PDAs, computers, home appliances, etc. These devices are distributed over different locations, at home, in the office, or in the car. Some devices may have one or more wired or wireless interfaces. The PN should support ubiquitous connectivity via infrastructure-based or ad hoc networks and integrate the heterogeneous devices in a seamless way. Moreover, the devices may have different resource constraints regarding energy supply, memory, and processing capability.
- **Personalization:** PNs are user-centric. For example, many people would like their favorite program to show up immediately when they turn on the TV. When the phone is ringing, they would like the volume of the TV to be automatically turned down. When they roam from one room to another, their favorite program should follow them on the displays that are presenting different rooms. Already now many devices can be programmed to suit personal preferences. We expect that future personal devices will be able to understand a person's needs, be aware of what he wants to do and try to accommodate the person's likes, habits, and situations. Therefore, a PN should offer personalization by understanding a person's needs; e.g., it should ensure that a person receives the right information at the right time in the right way.
- **Autonomy:** Autonomy has two aspects. The first one is that the devices constituting a PN, should be able to organize themselves with minimal human intervention. A PN consists of multiple clusters [9] which are geographically distributed, at home or wherever the person happens to be. The cluster topology is dynamic in the sense that nodes can join or leave a cluster, clusters can merge, or a cluster can be split up. These dynamics requires that the PN can discover these changes, self-organize components of clusters, and maintain the services or devices automatically. The second aspect of autonomy is that PNs should make their own decisions with minimal intervention by the user.
- **Security:** PNs are vulnerable to attacks by malicious intruders. The concept of a PN will only inspire trust and be accepted by its users when a sufficient level of security is guaranteed. Since PNs are centered on the needs of an individual, measures will have to be taken to protect the privacy of the owner. It requires that a PN is able to detect and react to attacks in a timely fashion. When an attack has happened, the PN should find suitable software to directly respond to this attack, update security software automatically, and filter out all packets coming from the origin address of this attack.

The characteristics outlined above require some kind of intelligent behavior in a PN. Under intelligence, in this context, we understand the capability to gather information about the personal environment, to learn from the information, from the prior knowledge of the environment, and from personal preferences, to adapt to personal preference automatically, to reason under uncertainty, and to react in advance.

4 The Capabilities of Cognitive Architecture for Personal Networks

In order to provide intelligent behavior in a PN, we need to design a cognitive architecture, a structure that forms the framework for the immediate processes of cognitive performance and learning [10]. The cognitive architecture should describe the detailed mechanisms on how cognition functions are implemented and how actions are performed. Therefore, the cognitive architecture of a PN concerns 1) the ability to access as many sources of knowledge as possible, 2) the ability to select proper AI techniques for the obtained knowledge, 3) the ability to make right decisions to actions, and 4) the ability to achieve its goals. Figure 2 illustrates the relationship between the cognitive architecture, knowledge, AI techniques, actions, and goals. We will discuss each of these in details as follows.

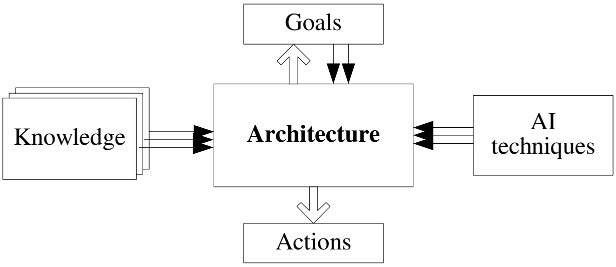


Fig. 2. The relationship between architecture, knowledge, AI techniques, goals and actions

- **The capability to manage knowledge:** Learning is a process of knowledge transfer. Therefore, the cognitive architecture should be able to manage all sources of knowledge of PNs as efficient as possible. Generally, there are three different sources of knowledge from PNs. First, knowledge about environments of PNs can be obtained directly from local personal devices, e.g., from sensors on a person's movement and on temperature etc. Second, some knowledge may not be obtained directly from external sources, but from internal sources generated by problem solving, such as a person's destination predicted from the person's movement. Third, some knowledge may be obtained from communication of distributed personal devices over a PN due to its distributed environment as mentioned before. Moreover, some knowledge may provide the whole information of a personal environment, some may carry the inaccurate or partial information of the personal environment, and some knowledge may change over time under a dynamical environment. These characteristics put forward a challenge on the design of a cognitive architecture for PNs. Therefore, the cognitive architecture should be able to not only efficiently access to and organize the knowledge, but also deal with uncertainty knowledge, such as noisy data and dynamical environments. The architecture should be able to facilitate component interaction in order to exchange data or access each other's services.

- **The capability to select and use proper AI techniques:** There are many existing AI techniques and algorithms developed on learning, reasoning and planning. Generally, the cognitive architecture is to perform a particular task using a proper AI technique with the knowledge obtained. Therefore, instead of having one-for-all solution, the architecture should be able to choose among the AI techniques that are appropriate for various types of learning tasks and different knowledge obtained. For example, the architecture selects a learning algorithm to learn knowledge about the past (like a user's routine) or a reasoning algorithm to infer the future situation based on the current knowledge. The architecture should be able to decide which algorithm is good for the limited/enough knowledge obtained and evaluate how the performance varies with the amount of knowledge.
- **The capability to make decisions:** The architecture is not only able to perceive its environment through sensors, but also able to act upon the environment through actuators. This is exactly as the same procedure as human being. When he sees something to touch, he needs to stretch out his hands to do so. The decision made by the cognitive architecture should be rational, which includes three aspects. First, it is able to do the right thing, for example, sending message, playing music, or displaying something on the screen. Second, it is able to trigger a right personal device. Third, if the personal device is remotely available, the architecture is able to detect the availability of resources for communications with the remote device. Given the knowledge obtained and proper AI algorithms used, the architecture first infers what actions can be made, then calculates the utility to be expected on taking each alternative according to the probabilities of the user's expectation and devices available to the user, and finally selects the alternative with the highest expected utility. Right decision should be made at the right time. In same cases, the architecture should be able to optimize when to make the decision considering resource costs, communication delay, etc.
- **The capability to achieve its goals:** Learning arises from goal-oriented activity. The architecture should have the ability to bring to bear all the knowledge that it has in the service of its goal. We envision that the goals of the cognitive architecture for PNs are to provide personal service and network management. Personal services means not only to specify the preferences of users, but also to infer similar services a user may have encountered or from uncertainty. Network management services provide personal networks to manage themselves. Network management includes fault management, performance management, configuration management, and security management. In order for the architecture to maximize the performance measure, the architecture might need to take some goal information to acquire and revise the knowledge through learning or to combine the knowledge obtained to make a decision. Imagine configuring the network at the destination when a user moving. The goal is to make sure the network is ready when the user reaches the destination. The architecture needs to acquire the knowledge on where the use is, what the destination can be, what the network he needs to use, and how long it would take to configure it. Taking these factors into account, the architecture finally makes the decision.

So far we have highlighted the main capabilities the cognitive architecture should support in a personal network environment. In order to achieve these capabilities, there arise two major challenges in designing the architecture. One is the interaction issue, including two aspects. The interaction between the architecture and the environment should be as flexible as possible in order to acquire as many sources of knowledge as possible and manage personal devices as efficiently as possible. The interaction between the architecture and the user should be as easy and convenient as possible to operate. The other is that the architecture should be open to be extended and modified easily with newly emerging functional entities.

5 The Cognitive Architecture of PNs

As illustrated in Fig.3, the proposed cognitive architecture consists of three layers: *the device layer, the cognitive layer, and the service layer. The device layer* provides context information to the cognitive layer from heterogeneous devices (such as sensors, mobile, cameras, and computers etc.). It also takes actions invoked by the cognitive layer. *The cognitive layer* is responsible for obtaining context data from the device layer and to deal with uncertainties to better serve the current situation of the owner of a PN. It consists of five components, namely *context cognition, personalization cognition, resource cognition, network cognition, and cognition management. The service layer* uses information (e.g., reasoning and inferring results) from the cognitive layer to adapt PNs for providing two kinds of services: personal services and network management. Let us discuss each of these components.

5.1 The Context Cognition

Context has been defined in many publications (e.g., in [11][12]). A primary functionality of the existing context awareness system is to sense both the state of the resources internal to the devices (e.g., the amount of memory, energy level of batteries) and external resources (e.g., location, velocity, available services in the proximity of the user, and user activity). We extend this functionality with the capability of learning by generating rules from the context history, reasoning under uncertainty, and providing context information to other components. For example, suppose that, most of time when one comes to the office, one turns on the light, the heating and computer and then reads. However, this is not always the case. Sometimes, one reads emails with a cup of coffee. Sometimes, one comes to the office just to pick up something one forgot. This presence of uncertainty radically changes the way the context cognition makes decisions. It needs to infer from these uncertainties, make a rational choice and act under uncertainty. As illustrated in Fig. 3, the work flow of the context cognition consists of three steps. First context data is collected from the context data provider in the device layer (e.g., from a sensor network). Then the collected data are reasoned about to deduct context information, such as the user's activity based on his movement. Finally, context information is generated and presented to actors in the device layer or other components of the cognitive layer.

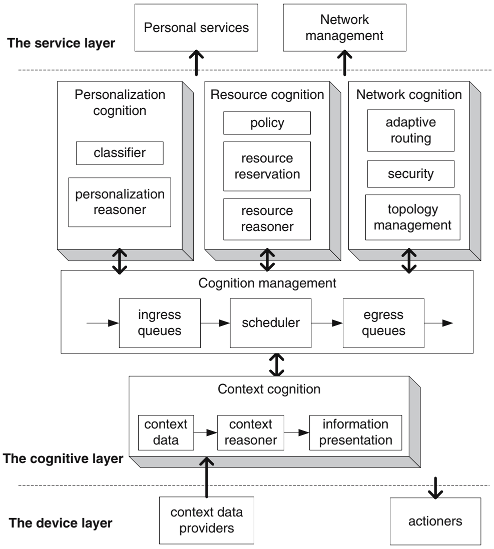


Fig. 3. The cognitive architecture overview

5.2 The Personalization Cognition

The personalization cognition's task is to build a generic model by understanding the needs of each individual. Personalization cognition is responsible for providing two functionalities: classification and personalization reasoning. The classifier examines incoming data from the context cognition and classifies the data into a set of personal profiles. This can be based on a system of rules. We provide a method to generate a user profile and classify the user profile data for personalization. The general structure of the method for classification is in the following:

DEFINE PROFILE NAME AS

((attribute [operator] value)

OR (attribute [operator] value)

AND (attribute [operator] value)

...)

The attributes are typically location, activity, person, and time. The operators are typically =, !=, >, <, >=, and <=. A number of (attribute [operator] value) sets is connected by OR or AND operation. Let us give examples of such a set of rules:

- *(Location = office) AND (Activity = come in) AND (Time >= 9:00AM) -> office manager.*
- *(Person ! = hosts) AND (Location = in front of the door) OR (Activity = knocking at the door) -> Guest assistant.*
- *(Sender = Maria) OR (date received <= today) AND (subject = appointment) -> Personal scheduler.*

The personalization reasoner mainly deals with learning personal preferences, reasoning about personal preferences to build personal profiles, and invoking actors in the device layer. Figure 4 illustrates an example of using a Bayesian network [13] to infer personal preference for a home assistant PN application. In this model, we describe six types of services: data services, telephone services, audio services, video services, diet-related services, and monitoring and controlling services. Each service may trigger different devices involved. If some services can be spontaneously used together, there is a directed link between them. Suppose that while watching TV, one likes to enjoy a cup of tea or juice, so, in this model there is a directed link from TV to either the tea or juice maker. The set of directed links or arrows specify the conditional dependent relationships. They are independent if there is no directed link between nodes. For example, we represent that time is independent of location by the absence of a directed link. Location has a directed link to phone services or data services and some other nodes. We call location the parent of these nodes. When one uses a phone service, one may choose a mobile phone, or a cordless telephone, or a PC to communicate. They are conditionally independent given the use of phone services. Using the Bayesian network, we can calculate the full joint probability distribution of each entry from the conditional probability distribution of each node given its parent.

5.3 The Resource Cognition

Resource cognition has three functionalities: policy generation, resource reservation and resource reasoning. The resource reasoner keeps track of each device by monitoring its resource constraints and presenting a reasonable estimate of its status. As mentioned before, a PN consists of a variety of devices with different resource constraints: energy rich or energy poor, nomadic or fixed, etc. Knowing these constraints is very important in selecting a node as an intermediary in adaptive routing or as a platform to run a particular application. For example, to provide high quality media streaming for entertainment, sensors cannot be selected as intermediaries because of their energy poverty and low processing capability. If a node is frequently moving from one place to another, it is not a good candidate for an intermediary either. Figure 5a depicts that the resource reasoner gains knowledge of each device based on its five resource constraints (energy level, memory, processing capability, bandwidth, and mobility) and produces a reasonable estimate of its possible status. The resource reservation reserves the devices based on the estimates given by the resource reasoner and releases the devices afterwards. We define three types of device states: idle, reserved, and busy. Figure 5b depicts a state diagram of resource reservation. The policy generation defines and generates a set of rules for resource reservation, for example, the small-world policy, where devices have only very few relations to a large number of other devices but have a lot of contacts with a limited number of devices. This is similar to one's address book, where one has a lot of names in the list. Most of them contacts are rarely sought, but with some contact sought very often. The small-world policy can provide priority information of each device for resource reservation and enable resource cognition to easily reserve the resource of a proper device.

We see that resource cognition works as *a knowledge gainer* of a network and as *a controller* to network cognition.

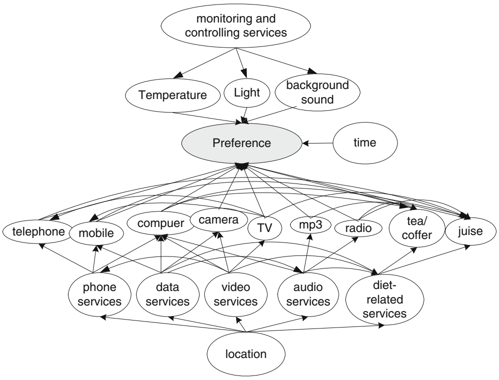


Fig. 4. An example of a Bayesian network for inferring personal preferences

5.4 The Network Cognition

The primary goal of network cognition is to enable the network to organize itself, detect problems, repair or recover itself, and optimize itself. Network cognition mainly consists of three functionalities: adaptive routing, security management and topology management. Adaptive routing uses the feedback information from resource cognition, plans a reasonable route and optimizes the route for performance or reliability. Figure 6 illustrates an example of a cumulative learning process for adaptive routing based on the feedback from resource cognition. Consider the following example. Suppose the case of a node A sending packets to a node B via a node C. On knowing that node A needs to send to node C first, it concludes that all packets destined to node B need to be sent to node C. Yet when it knows that node C has been moved out of its (radio) range, which is a feedback from resource cognition, it does not consider node C any longer as an intermediary to node B. The feedback from resource cognition allows node A a faster (e.g., real-time) adaptation, thus preventing unnecessary traffic. Security management should detect possible attacks in real-time, perform vulnerability assessments, and forecast threatening tendencies. Compared to traditional security management which reacts to an attack and eliminates the consequences, this security management enhances this strategy with forecasting and

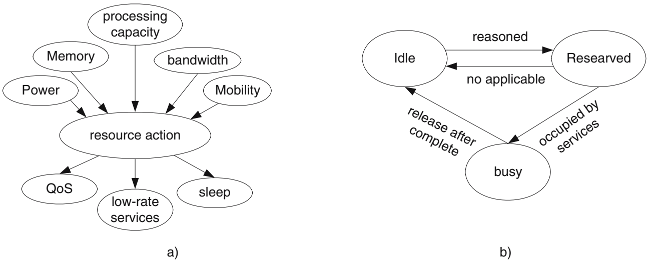


Fig. 5. a) Reasoning from resource constrains for resource reservation. b) State diagram of resource reservation.

prevention. In case the attack is not prevented, the second react strategy is used to find suitable software to update the system and the system should learn from this experience. Topology management manages the dynamics of a PN, e.g., nodes leaving or joining a cluster. Topology management uses cognitive monitoring to collect data on the current situation of the network and to predict the future situation of the network. It speeds up the adaptation of the network's behavior to accommodate changes and achieves a more efficient use of resources in controlling the potentially large dynamics of the PN.

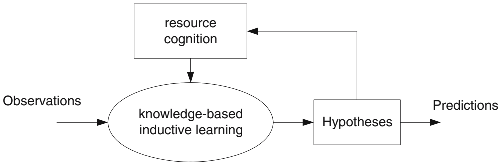


Fig. 6. A cumulative learning process for adaptive routing

5.5 Communications: The Cognition Management

The four above mentioned cognition components can work alone for an application. But most often, they need to work together. Cognition management is responsible for the coordination and the interaction of the four cognition components. It consists of an ingress queue, a scheduler, and an egress queue. The scheduler schedules the requests or responses to the cognition components, such as to context cognition, or to resource cognition. In order to guarantee real-time requirements, the scheduler is based on a priority queue and priority threads. The cognition management can also be used by the cognitive layer to discover its peers, to negotiate and to co-operate with them since they are distributed over a PN. For example, a context cognition entity in one place can send its context information to a resource cognition entity in another place where the node is moving to and to reserve resources there. The advantage of using cognition management is that it enables the whole system to be modular and extendible in the sense that the cognition components can be replaced or updated independently without impacting the other components. Further, cognition components can be added to this layer without impacting the integrity of the whole architecture.

6 Services Scenarios

In order to demonstrate the intended functionality of the proposed framework, in this section, we illustrate some service scenarios based on this architecture.

Figure 7 demonstrates how the cognitive architecture is used to manage session mobility (sessions moving between personal devices) in PNs. The user is using his computer and an attached microphone and speaker to have a video conference (which includes a video and audio session, and might contain a data session for text chat.) with his team workers in his office. Around 5:00PM, his personalization cognition reminds him that he needs to pick up his child from school. The context cognition in the office discovers that his mobile with an embedded camera can be available for the continuity of the video conference sessions. The cognition management sends a handoff query to the mobile. The resource cognition of the mobile reasons about the possibility of available resources for the handoff and reserves the resources. The network cognition of the mobile binds the sessions with their network interfaces. Finally, the mobile sends a handoff notification to the computer with the IP address and port numbers of the network interfaces. Before he leaves his office, the sessions have been merged (from three devices to one device) into his mobile assuring continuity. When he is on the move, the context cognition in his mobile predicts that the user is going to his car. The cognition management of the mobile sends again a handoff query to the cognition management of the car. The context cognition of the car discovers that either his mobile or the GPS display with a Bluetooth camera and headset are available for the continuity of the sessions. The resource cognition reasons that the mobile might run out of the battery power and concludes that the GPS display with the camera and the headset is better for providing the service. So the resources of the GPS, the camera and the headset are reserved and the network cognition of the car cluster binds each session with its IP address and port number of the device. Thus the sessions have been split across these three devices and the network cognition manages the communications of these devices. When he is driving home, the personalization cognition of the home cluster reasons that the user would like to use his TV for the video service and his stereo system for the voice service. In the same way, the handoff query is negotiated and the relevant resources at home are reserved. In addition, the personalization cognition at home infers the user might need a cup of coffee. So the coffee machine is ready. Meanwhile the heating is turned to his favorite temperature. When he reaches home, the light is on and the sessions have been migrated to the TV, the webcam and the stereo system.

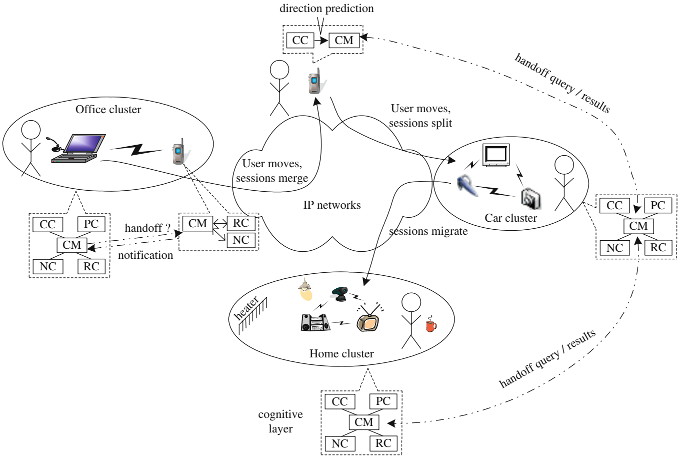


Fig. 7. A scenario of session migration in personal networks

From the above scenario, we can see that by using cognitive techniques, the proposed architecture enables proactive configuration and adaptation before session migration, merge, and split across devices. Many existing systems suffer handoff delays due to too slow configuration speed (e.g., device start-up and high setup latency). This is exacerbated when stringent security requirements and QoS come into play. In particular for personal applications the maximum allowed configuration time, e.g., to get a device that is turned on to become part of the PN is determined by the patience of the user (often of the order of seconds). The cognitive approach hides the latency by predicting when a configuration will be needed and setting up available resources in advance without the user or applications being aware of this.

7 Research Issues on the Cognitive Architecture Implementation

When implementing the cognitive architecture for PNs, the following issues need to be addressed.

- How are functional entities of the cognitive layer mapped on personal devices of a particular domain? From a device point of view, due to the heterogeneity of personal devices, some devices may be capable of supporting full functional entities, some may not. From the functionality point of view, it may not be necessary to implement all entities in each device. Therefore, whether it is highly centralized or fully distributed in one domain is one of the considerations.
- If it is fully distributed in one domain, what can the minimal set of functionalities be installed on a device? How are the devices capable of providing adaptation in terms of modification of existing features or adding new ones?
- Whether the implementation of the cognitive layer is hidden from the service layer or not is a concern. Hiding all the details of the implementation will lead to a very large software code, which is often considered difficult to construct and maintain. Transparency of some functionalities to the service layer will lead to a lightweight implementation in terms of computational complexity and amount of software code. However, it will raise the question on how the service layer can monitor and adapt the behaviors of the cognitive layer according to its needs.
- How smart should the cognitive architecture be? On the one hand, a proactive system can enhance productivity, safety, awareness, and efficiency [14]. On the other hand, a proactive system can also annoy a user and thus defeat the goal.
- How fast are learning algorithms? In order to learn from a person's preferences and to make decisions, the person's behavior needs to be observed and a hypothesis representation needs to be generated from the data. How much observed data is sufficient? How accurate is the hypothesis representation and how much time is required?
- How efficient can the architecture be in managing distributed information over PNs? Since cognition components in the cognitive layer are distributed within a PN environment, data collection and dissemination is a challenge. Can the interaction of the cognition components be tuned to achieve optimal performance? Does the cognition management work efficiently among its peers?
- What the cost of implementing this new functionality on these devices is also a major concern for consumers.

Solutions to these issues should be researched and developed in the future. But at first, we will map proposed reasoning and learning functionalities on personal devices and investigate their efficiency and performance. Meanwhile, we will explore more possible cognitive techniques for personal networks.

8 Conclusions

In this paper, we have explained and given a motivation of why cognitive networking is needed in personal networks. We proposed a cognitive architecture which explores how cognitive technology could be applied into PNs for both personal services and network management. We illustrated some examples which can benefit from using the cognitive techniques and the proposed architecture. This architecture develops a framework for pervasive computing applications and enables distributed intelligent processes to manage personal networks. Finally, we outlined some research issues on implementation of the architecture.

Acknowledgment

The work presented here was funded by the Dutch Ministry of Economical Affairs under the Freeband PNP2008 project. This work expresses the view of the authors and not necessarily the general view of PNP2008.

References

1. Niemegeers, I., de Groot, S.H.: Research issues in ad-hoc distributed personal networking. Special Issue of the journal Wireless and Personal Communication 26(2-3) (2003) 149 - 167
2. Weiser, M.: The computer for the 21st century. Scienti_c American 265(3) (1991) 94 – 104
3. Clark, D.D., Partridge, C., Ramming, J.C., Wroclawski, J.T.: A knowledge plane for the internet. In: SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications. (2003) 3 – 10
4. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Computer Society 36(1) (2003) 41 - 50
5. Dietterich, T., Langley, P.: Machine learning for cognitive networks: Technology assessment and research challenges. Technical report, Oregon State University (2003)
6. (Carnegie mellon university project aura) http://www.cs.cmu.edu/ aura/.
7. (Microsoft research project easyliving) http://research.microsoft.com/easyliving/.
8. (Mit project oxygen) http://oxygen.lcs.mit.edu/Overview.html.
9. Jacobsson, M., Niemegeers, I.: Privacy and anonymity in personal networks. In: Proceedings of the 2nd International Workshop on Pervasive Computing and Communication Security (PerSec'05). (2005)
10. Newell, A.: Uni_ed Theories of Cognition. Harvard College (1990)
11. Schilit, B., Hilbert, D., Trevor, J.: Context-aware communication. IEEE Wireless Communications 9 (2002) 46 - 54
12. Abowd, G.D., Anind K. Dey, e.a.: Towards a better understanding of context and context-awareness. In: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing. (1999) 304 - 307
13. Russell, S., Norvig, P.: Arti_cial Intelligence: A Modern Approach. Prentice Hall (2003)
14. Kahn, K.C., Culler, D.E.: Ad hoc sensor networks: Proactive computing is coming. Technical report, Intel Corporation (2002)

A Cross-Layer Architecture for Autonomic Communications

M.A. Razzaque*, Simon Dobson, and Paddy Nixon

Systems Research Group
School of Computer Science and Informatics
UCD Dublin IE
abdur.razzaque@ucd.ie

Abstract. Layered architectures are not sufficiently flexible to cope with the dynamics of wireless-dominated next-generation communications. Most existing architectures and approaches depend purely on local information and provide only poor and inaccurate information gathering at the global scale. Delayered or cross-layer architectures may provide a better solution: cross-layering allows interactions between two or more non-adjacent layers in the protocol stack. We propose a new cross-layer architecture which provides a hybrid local and global view, using gossiping to maintain consistency. We evaluate our proposal informally in terms of communication complexity and in terms of its ability to support the "self-*" properties being proposed within the autonomic communications community.

Keywords: Cross-Layer Architecture, Autonomic Communications, Gossiping.

1 Introduction

The worldwide success of the Internet is dominated by the layered architecture, but a strict layered design is insufficiently flexible to cope with the dynamics of wireless-dominated next-generation communications. Recent studies [1] show that careful exploitation of some cross-layer protocol interactions can lead to more efficient performance of the transmission stack – and hence to better application layer performances – in different wireless networking scenarios.

Cross-layer design breaks away from traditional network design, where each layer of the protocol stack operates independently and exchanges information with adjacent layers only through a narrow interface. In the cross-layer approach information is exchanged between non-adjacent layers of the protocol stack, and end-to-end performance is optimized by adapting each layer against this information Cross-layering is not the simple replacement of a layered architecture, nor is it the simple combination of layered functionality: instead it breaks the boundaries between information abstractions to improve end-to-end transportation.

* This work is partially supported by Science Foundation Ireland under grant number 04/RP1/1544, "Secure and Predictable Pervasive Computing."

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 25 – 35, 2006.

One obvious shortcoming of the strict layering is the lack of information sharing between protocol layers. This hampers optimal performance of the networks, since shared layer information is the prerequisite for many forms of performance optimization. On the other hand, cross-layer systems shift the research landscape away from optimizing the performance of individual layers, and instead treat optimization as a problem for the entire stack. The technique consists of taking into account information available from different levels [2], not necessarily adjacent, in order to create a system much more sensitive to its environment, load and usage. Essentially we replace the narrow inter-layer interface with a wider, richer and more holistic view of the network's issues, goals and constraints.

The assumptions in the wired IP stack are inadequate for wireless networking, and TCP/IP is known to suffer from performance degradation in mobile wireless environments. This is because such environments are prone to packet losses due both to high bit error rates and mobility-induced disconnections. TCP interprets all packet losses as an indication of congestion and (inappropriately) invokes congestion control mechanisms, which leads to degraded performance. Cross-layering between Link Layer and Transport Layer is an easy solution to this problem [3].

On the other hand, it is clear from the recent initiatives in autonomic computing and autonomic communications [4, 5] that there is a need to make future networks self-behaving, in the sense that they work in an optimal way with "endogenous" management and control, with minimum human perception and intervention. Autonomic communication is the vision of next-generation networking which will be a self-behaving system with properties such as self-healing, self-configuration, self-organization, self-optimization and so forth – the so-called "self-*" properties. To attain this self-behaving system within existing strictly-layered approaches may be possible to certain extent, but will not (we claim) leverage all the possible optimizations. We consider cross-layer architectures to be better suited to achieving the self-optimization, self-configuration and other "self-*" properties targeted by autonomic approaches.

Most existing architectures (including GRACE [6], WIDENS [7], MobileMan [8]) rely on local information and views, without considering the global networking context or views which may be very useful for wireless networks in optimizing load balancing, routing, energy management, and even some self-behaving properties like self-organization. Only CrossTalk [9] is based on a (even partially) global view of the network. On the other hand, POEM [10] is the only architecture considering self-optimization that could be helpful for autonomic communication. Collecting and maintaining network-wide, global statistics can be expensive, while global actions are hard to co-ordinate; however, the effects of such systems can often be dramatic, and they can address problems that are difficult to detect, diagnose or solve using purely local information. To explore the impact of this idea, we propose a new cross-layer architecture based on building and maintaining a global view of the network's state and constraints, utilizing gossiping for gathering information from neighboring nodes. For scalability and overhead issues, we limit gossiping processes among direct neighbors.

Section 2 describes related study on cross- layer architectures. Our contribution, a cross-layer architecture for autonomic communications is presented in section 3.

Section 4 briefly describes the difficulties implicit in cross-layer interactions. And section 5 concludes with some directions for future work.

2 Related Study

Research on cross-layer networking is still at a very early stage, and no consensus exists on a generic cross layer infrastructure or architecture. However, the importance of a sound architecture to handle the proliferation of cross-layer operations in wireless as well other communications media is clear, especially in autonomic systems for which properties need to be specified and maintained with minimal manual configuration and intervention [11]. A number of proposals for cross layer designs and their corresponding architectures have been published in the literature. Most of these architectures are relying on node-local view and very few utilize both the local view as well as network-wide global view.

GRACE (Global Resource Adaptation through Co-opEration) [6] is a cross layer adaptation framework. All system components (hardware, network, and operating system) and applications are allowed to be adaptive. These adaptive entities co-operate with each other to achieve a system-wide optimal configuration, for example to maximize system utility, in the presence of changes in the available resources or application demands. However, its cross-layer approach includes no explicit consideration of cross layering within the networking layers or protocol stack. WIDENS (Wireless DEployable Network System) [7] has been proposed with an aim to acquire the interoperability, cross layering and re-configurability at the same time. This cross layering architecture seems a promising one where protocol optimization is based on the local state information but it is still in the validation stage and so lacks any real measurement of efficiency especially in terms of performance.

The MobileMan [8] architecture presents, along with the strict layering, a core component, *Network Status*, which functions as a repository for information that uniformly manages the cross-layer interaction while respecting the principle of dividing functionalities and responsibilities in layers. The approach aims to optimize overall network performance with respect to local state information by increasing local interaction among protocols, decreasing remote communications, and consequently saving network bandwidth. Performance improvement verifications are yet to be published. ECLAIR [12] is local-view-based, efficient cross-layer architecture for wireless protocol stacks. Along with legacy protocol stack it consists of two main components: an *Optimizing Sub-System*, the cross layer engine which contains many *Protocol Optimizers*, which are the “intelligent” components of it, and *Tuning Layers* provide the necessary APIs to the protocol optimizers for interacting with various layers and manipulating the protocol data structures. There is no processing overhead on the existing stack since the optimizing subsystem executes in parallel to the protocol stack.

CrossTalk [9] is the only (as far as we are aware) cross-layer architecture, which has the ability to reliably establish a network-wide, global view of the network under multiple metrics. Having such a global view, a node can use global information for local decision processes in conjunction with a local view containing node-specific information contributed by each layer of the stack or system component. To keep overheads low, no additional messages are sent: instead the local information taken from the local view is piggybacked onto outgoing packets. Piggybacking implies that it is quite unlikely that any node will obtain fully accurate global view under many likely models of data exchange. Even with an uncertain and poor global view, however, CrossTalk has shown performance improvement in a load balancing algorithm specifically reducing per-hop packet delay and bottlenecks. It seems reasonable to expect such performance to be improved by improved global modeling of the network and this expectation is the encouragement for the new cross-layer architecture.

Researchers have addressed the importance of "knowledge plane" or "knowledge network" for the adaptability or context awareness in next generation communications. Clark *et al* [13] describe the Knowledge Plane as a new communication paradigm where a network can assemble itself given a set of high-level instructions and constraints, re-assemble itself as requirements change, automatically discover when something goes wrong, and automatically fix a detected problem or explain why it cannot do so. This AI-based approach considers an additional network layer between the network and the application layer, as the place in which nearly all network control activities take place. Mulvenna and Zambonelli [14] present an abstract architecture for knowledge networks that addresses the key issues of how both physical contextual knowledge and social knowledge from the users of communication networks can be used to form a knowledge space in support of autonomic agents dealing with network elements and applications. To avoid the burden of an additional distributed computational layer, and to more successfully promote cross-layer interactions, they consider knowledge networks as managed by existing components at the application and network levels. Both proposals are at the abstract level: still, these are inspiring issues for the use of a Knowledge Plane in our architecture. However, our cross-layer, architecture-oriented knowledge plane approach is somewhat different than these approaches.

It is possible to utilize cross-layer information to attain some of the self-behaviors of autonomic communications mentioned earlier. POEM (Performance-Oriented Model) [10] is perhaps the first initiative towards developing a cross-layer based self-optimizing protocol stack specifically for autonomic communication. For the optimization purposes it utilizes local state information. The basic design criterion is self-optimization is a control-plane issue where the normal functions of the protocol stack should not be compromised, with added cross-layer benefits being layered on top. The system is being investigated both formally and through simulation. In their patent filing, Sadler *et al* [15] link cross-layer information to self-healing. In this architecture, a Management Plane exists along side of the protocol stack to stores information obtained from each protocol layer in the Cross-Layer Platform and nodes use this information to help the routing protocol maintain network reliability in the presence of failures. Interestingly none of the self-* behaviors in autonomic computing and communication are highly orthogonal, which means there is some dependency between them – self-healing is partly supporting self-organization, and vice versa.

3 A Cross-Layer Architecture Based on a Global View

Both OSI and TCP/IP adopt a bottom-up approach driven by physical and network constraints, which can make it hard to capture and respond to top-down user demands or contexts. Cross-layer design can help capture these concerns by providing a more uniform framework within which to capture and disseminate information at different semantic levels. Realizing the importance of cross-layering – and specifically cross-layering architectures with a network-wide global view – in next-generation communications, we propose an alternative cross-layering architecture based on a combination of node-local and network-global views. “Global view” is a broad term suggestive of centralization and reduced scalability, but we will try to establish and maintain a network-wide view of multiple metrics depending on the focus of the network without undue impact. Metrics such as load, battery status and so on in can be very useful in attaining self-organization or self-healing in an autonomic network, and having a global view allows a node to evaluate its own status against the average within the network at any instant. For example, a node could decide whether it is carrying more traffic then the average node and how overloaded it is compared to the average, and could utilize this information for routing and load balancing.

As this architecture is based on local information as well as global and the sources of information are different, we require two different schemes to gather information. This leads to the use of a knowledge plane consisting of two entities to manage information efficiently. One entity is responsible for the organization of locally available information from different layers in the local node’s network stack, provided by each layer of the protocol stack independently: battery status, load, neighbor count, signal-to-noise ratio transmit power, bit error rate, velocity and so forth. Each protocol layer can access this data and use it for local optimizations. The sum of this information represents the state of the node or *local view* on the network. The other data-management entity establishes a network-wide or *global view* summarizing information of the same types collected in the local view. To produce the global view we use an information dissemination procedure based on gossiping.

3.1 Motivation of This Architecture

Why another architecture? We believe that existing approaches contain some excellent approaches to particular problems in the construction of autonomic networks with cross-layer optimization: however, there is also only a fairly weak integration of the various techniques into a systems-level view of cross-layer interactions.

Cross-layer interactions can be managed by information exchange between layers, or through a more structured knowledge plane approach. Per-layer interactions are potentially more efficient, but also expose significantly more design information to the individual layers, which can lead to unnecessary coupling between implementations. It also tends to bind information to its source, in the sense that a layer will always have to commit to the layer originating information it requires for optimization. A knowledge plane decouples information from its source, abstracts the internal designs of layers and provides a central facility within which to provide reasoning and other advanced features.

Self-* properties are almost always phrased in whole-network terms: self-optimization, for example only makes sense if the network has a global property to optimize against, as individual local optimizations are unlikely in general to converge to a global optimum. A naïve implementation of a knowledge plane would inevitably be a performance bottleneck. Maintaining a global view without introducing performance and reliability concerns implies constructing the global view in a distributed manner, accepting the inevitable problems with latency and inconsistency.

Maintaining a distributed representation of knowledge means either locating some of the information at a particular node, or replicating the entire knowledge plane on each node, or a combination of the two. Since we do not want to over-commit to a particular strategy, we use a gossiping approach coupled with information summarization and fusion to maintain local copies of the knowledge plane. The local component acts as a gateway to the complete knowledge base, containing both local and summarized global views important metrics. A client layer need not know the source of particular information, and tailoring the gossiping algorithm provides a single point for handling robustness and latency issues. Using randomized selection of nodes with which to gossip, for example, produces a random communication structure that behaves rather like noise and does not introduce hot-spots or other artifacts. We conjecture that the overheads of such a scheme will be greater than the piggy-backing used in CrossTalk but significantly less than in deterministic flooding [16].

3.2 Overview of the Architecture

Alongside the existing layers, the Knowledge Plane is the key element of the architecture. Direct communication between layers and a shared knowledge plane across the layers are the two widely used cross-layer interactions polices. Because of the improved separation and management possibilities we prefer to utilize the knowledge plane for the architecture. The following are the main elements of the architecture:

Existing TCP/IP layers: These provide normal layering support when it is necessary, as well the information to the knowledge plane related to different layers to maintain local view of the node. This allows full compatibility with standards and maintains the benefits of a modular architecture, as it does not modify each layer's core functionality.

Contextors for different layers: Each layer in the existing protocol stack has a corresponding *contextor*, which will act as their corresponding interface between the layer and the knowledge plane. Each of these contextors acts as a "tuner" between a layer and the knowledge plane. Possible functionality for manipulating protocol data structures is built into the contextors: no modification is required to the existing protocol stack. This facilitates incorporation of new cross layer feedback algorithms with minimum intrusion. Generally a protocol implementation has data structures for control and data, with a protocol's behavior being determined by its control data structure. A contextor is responsible for reading and updating the control data structures when it is necessary.

Knowledge Plane: A common knowledge plane database is maintained to encapsulate all the layers' independent information as well as the network-wide global view, which can be accessed by all layers as needed. For modularity it maintains two entities responsible for maintaining the local and global views.

A knowledge plane can act in one of two ways: as a simple database with local and global view related information, or as a database with intelligence, which can manipulate its information to make inferences. The former allows each layer to provide whatever querying and optimization functions it requires, but forces all the complexity for information fusion into the contextors; the latter allows certain fusion and uncertain-reasoning functions to be encapsulated within the knowledge plane, making them available uniformly to the contextors and simplifying their local coding – at a cost of complicating the knowledge plane with the consequent risks of performance problems and feature interaction. There is clearly a trade-off in functionality that requires careful exploration.

Interaction between different layers and the knowledge plane through client programs can be reactive (responding to changing context) or proactive (anticipating changes and provisioning accordingly). Generally the interactions between different layers and the knowledge plane are event-oriented, which suggests a reactive scheme; on the other hand, the knowledge plane can maintain a model of the network and act autonomously to issue its own events. This leads to improved performance if the model leads to a correct proactive adaptation, but can be detrimental if the projection is wrong.

In our architecture we are considering the database with intelligence as shown in figure 1. The knowledge plane consists of the database and necessary optimizing algorithms. The database is separated into local view and global view for isolation and management purposes, although it appears unified to clients.

Gossiping: Gossiping is considered as one of the most promising data-dissemination mechanisms in peer-to-peer or distributed systems. There are number of algorithms that can be classified as *reactive*, *proactive* and *periodic*. As there are few comparative performance studies amongst these algorithms, it is difficult choose the most suitable algorithm for a particular purpose. In our case we propose a periodic gossiping approach, possibly with out-of-band “immediate” signaling for important changes.

The gossiping service is built on top of existing TCP/UDP, and is responsible for gathering information from other nodes to generate the network-wide view at the host node. At each exchange the gossiping service chooses another node in the system (either randomly or with some weighted preference) and exchanges its local state with that node. In this architecture we will consider a gossiping exchange as an application-level event which will trigger the knowledge plane to take the necessary actions.

3.3 Interactions Between Protocol Layers and the Knowledge Plane

Events and the state variables of different layers are the two most important concerns for the interaction between the layers and the knowledge plane. This interaction is the most delicate issue in a cross-layer architecture. There are two basic models:

- The knowledge plane registers with the contextors, and receives events whenever a change in the layer’s context occurs; or
- The knowledge plane periodically retrieves state information form all layers.

We will here consider only the first approach further, as we believe it is better-suited to autonomic systems. The interaction policy between the layers and the knowledge plane through the contextors is summarized below (and as shown in figure 1):

Step 1: A contextor attaches to the control data structure information of the appropriate layer.

Step 2: The database part of the knowledge plane registers with the contextors to receive change events.

Step 3: Contextors notify the knowledge plane's database about events (timeout, disconnections, *etc*) and pass the necessary information regarding these context changes, which are incorporated into the model. Optimization algorithms monitor the database waiting for "guard" predicates to become true, and are executed when their guards are satisfied.

Step 4: The algorithms access the model to acquire any additional information they need in order to make their decisions. Algorithms are not restricted in which parts of the model they may access.

Step 5: Control actions are propagated back through the contextors to the layers' control plane.

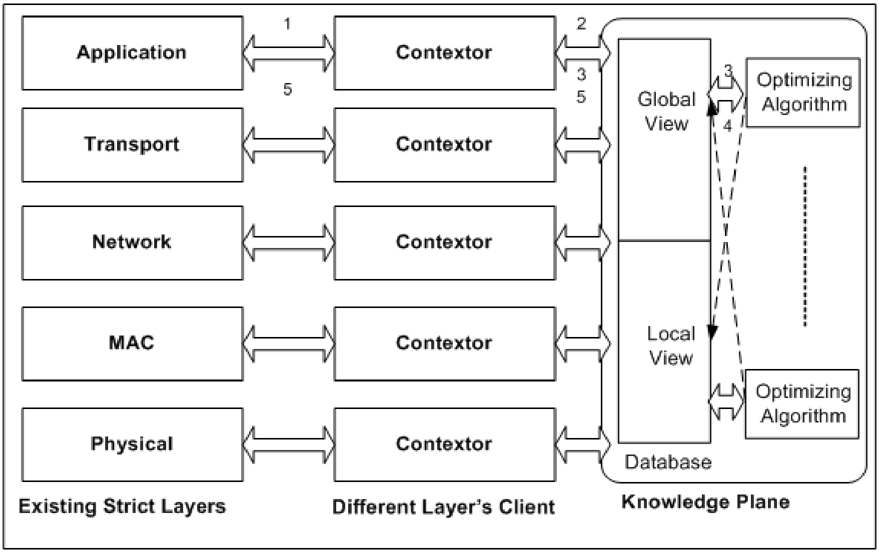


Fig. 1. Sample interactions between different layers and the Knowledge Plane

3.4 Application Scenario

Prospective application areas for this architecture could be optimization of load balancing, routing, energy management in wireless networking and obtaining self-behaving properties of autonomic communications like self-organization. All of these applications require some knowledge about their neighbors, which can be provided by our architecture.

As an application scenario, consider the case of a wireless network self-organizing to maintain communications between nodes. Self-organization can be defined as the emergence of system-wide adaptive structure and functionality from simple local interactions between individual entities [17]. An application scenario with 7 nodes is shown in figure 2. In scenario (a) node s has a request for node d1 and it is using the route s-n1-n2-d1. In this case, we will consider all the nodes are gossiping knowledge (the global view) about their primary neighbors, so node s has knowledge about n1, n1 has about n2 and n3 and so on. If after transmission begins n2 fails, existing (local-view only) routing protocols would have n2 receiving the packet, determining d1 to be dead, and finally sending a "node unreachable" error message to s which wastes all the resources committed to the exchange. Using a global view, however, if d1 and d2 are giving almost same type of services a suitable global view would allow n2 to determine that in case of d1's failure d2 can meet the request of s. This requires making information about the service-level capabilities of a node available to the routing layer, which is facilitated by our approach and can easily be expressed as an optimization algorithm. This leads to scenario (b) where nodes have re-organized because of the death of d1, and once n2 gets the request from s it reroutes to d2 instead of d1 and fulfills the request. With this action, our architecture can conserves energy and minimizes latency by eliminating the overhead required to invalidate the current route, establish a new route, and retransmit the request. Moreover, it can preserve the original route when failed node becomes available. Essentially we provide in this scenario a generalized form of content- or service-based routing, without committing extra specific resources or network topology to the task and so retaining complete freedom to adopt other strategies as required.

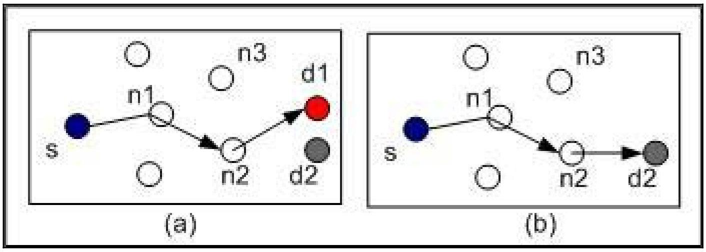


Fig. 2. Application Scenario

4 The Difficulties Implicit in Cross-Layer Interactions

Cross-layer design is not an easy task, as co-operation among multiple protocol layers has to be coordinated without leading to conflicts or (worse) loops. A common drawback is the lack of a systematic approach for cross-layer designs overall, not just its interactions: individual optimizations run at cross purposes, violating the structural architecture principles for only shortsighted performance gains, and could lead to serious consequences through unexpected feature interactions.

Once layering is broken, the luxury of designing a protocol in isolation is lost too. Unbridled cross-layer interactions can create loops, and from a control theory point of view become a hazard to the stability of the system. Loosely controlled interactions can also result in "spaghetti code," stifling further innovation and proliferation on the one hand, and increasing the cost for maintenance on the other. In severe case, the overall system will have to be redesigned. (These problems are detailed in [11] with examples.) At a critical time in the history of wireless communication, we need to note some of the adverse possibilities and exercise proper caution. Cross-layer design for new network architecture is the trend; however we think there are a few principles that have to be followed:

- Cross-layer design should be holistic instead of being fragmenting
- Layering should remain, for proliferation and longevity of the system
- Cross-layer interactions should be done in a controlled way, and preferably through a common optimization interaction interface
- Dependencies between the interaction protocol parameter have to be examined to avoid loops

Cross layering tries to share information amongst different layers, which can be used as input for algorithms, for decision processes, for computations, and adaptations. This process of sharing has to be coordinated and structured somehow since cross layering could potentially worsen the performance problem that it intends to solve. This is due to several effects. Optimization processes at different layers could try to optimize a common metric in opposite directions. Furthermore, two different metrics could have negative impacts on each other when trying to optimize them, such as energy efficiency and delay. A general problem is that altering a metric at one layer often has an effect on other layers implicitly. For example, altering the transmission power on the physical layer can have an effect on the network layer as nodes might disappear from the direct transmission range.

5 Conclusions and Future Work

The worldwide success of the Internet has led to the domination of the layered architecture, but a strict layered design is not flexible enough to cope with the dynamics of next-generation communications. Moreover exploitation of some cross-layer protocol interaction can lead to more efficient performance of the transmission stack (and hence better application layer performances) in different wireless networking scenarios. Most existing architectures depend on the local information and only CrossTalk depends on local as well as network wide global view. Even with the uncertain and poor global view gathering process, CrossTalk has shown performance improvement in load balancing algorithm. With a more accurate global view gathering process we can hope for further improvements, and with this in mind we proposed an alternative cross-layering architecture for autonomic communications which is based on local information as well as global information and gossiping will be the mechanism to collect the global view related information. The use of gossiping provides more impact that piggy-backing but keeps overheads more controlled than flooding, and allows the time constants and latencies of information to be varied widely.

Our proposed architecture is still in the design stage. We are currently engaged in exploring how we may provide cross-layer optimization of TCP/IP, simulating existing TCP enhancements and comparing them to a novel version whose adaptations are driven from a knowledge plane populated using our gossiping technique. This will allow us to evaluate both the gossiping approach and the various cross-layer parameters that can be used to influence TCP behavior, as well as comparing them against established approaches with less cross-layer influence.

References

1. Srivastava, V. and Motani, M, "Cross-Layer Design: A Survey and The Road Ahead", IEEE Communications Magazine, Volume 43, Issue 12, Page(s): 112 – 119, Dec. 2005
2. Simon Dobson, "Putting meaning into the network: some semantic issues for the design of autonomic communications systems", In Mikhail Smirnov, editor, Proceedings of the 1st IFIP Workshop on Autonomic Communications, volume 3457 of LNCS, Springer Verlag, 2005,pp: 207-216
3. H. Balakrishnan, "Challenges to reliable data transport over heterogeneous wireless networks", Ph.D. Dissertation, The University of California at Berkeley, USA, 1998
4. 'Autonomic Communication Forum', HYPERLINK "http://www.autonomic-communication-forum.org" http://www.autonomic-communication-forum.org
5. 'Autonomic Communications Initiatives' HYPERLINK "http://www.autonomic communication.org" www.autonomic-communication.org
6. Daniel G.Sachs et al, "GRACE: A Hierarchical Adaptation Framework for Saving Energy", ACEED 2005
7. Dzmitry Kliazovich and Fabrizio Granelli, "A Cross-layer Scheme for TCP Performance Improvement in Wireless LANs", Globecom 2004, IEEE Communications Society, pp. 841-844
8. Marco Conti, Gaia Maselli, Giovanni Turi, Sylvia Giordano "Cross layering in mobile Ad Hoc Network Design", IEEE Computer Society, pages 48-51, February 2004
9. Winter, R.; Schiller, S.; Nikaein, N.; Bonnet C., "CrossTalk: A Data Dissemination-based Cross-layer Architecture for Mobile Ad-hoc Networks", IEEE Workshop on Applications and Services in Wireless Networks (ASWN 2005), Paris, June 2005
10. X. Gu, X. Fu.H. Tshofenig, and L. Wolf, "Towards Self-optimizing Protocol Stack for Autonomic Communication: Initial Experience", Proceedings of the 2nd IFIP International Workshop on Autonomic Communication (WAC'05), Springer Lecture Notes in Computer Science Vol. 3854 (LNCS), October, 2005, pp: 186-201
11. V. Kawadia and P.R. Kumar, "A Cautionary Perspective on Cross-Layer Design", IEEE Wireless Communications, February, 2005, pp.3-11
12. V. T. Raisinghani and Sridhar Iyer, "ECLAIR: An Efficient Cross Layer Architecture for Wireless Protocol Stacks", WWC2004
13. D. Clark et al., "A Knowledge Plane for the Internet", Proceedings of the 2003 ACM SIGCOMM Conference, Karlsruhe (D), ACM Press, 2003
14. M. Mulvenna, F. Zambonelli, "Knowledge Networks: the Nervous System of an Autonomic Communication Infrastructure", 2nd IFIP Workshop on Autonomic Communication, LNCS No. 3854, 2006
15. C. Sadler, W. Chen, and L. Kant. , "Cross-Layer Self-Healing in a Wireless Ad-Hoc Network", U.S. Patent filed April 2005
16. M. Lin, K. Marzullo, S. Masini, "Gossip versus deterministic flooding: Low message overhead and high reliability for broadcasting on small networks", UCSD Technical Report TR CS99-0637
17. Prehofer, C.; Bettstetter, C**.,** **"**Self-organization in communication networks: principles and design paradigms", IEEE Communications Magazine, July 2005 Page(s): 78 - 85

Self-configuration of Network Devices with Configuration Logic

Sylvain Hallé, Éric Wenaas, Roger Villemaire, and Omar Cherkaoui

Université du Québec à Montréal
C.P. 8888, Succ. Centre-ville
Montréal (Canada) H3C 3P8
halle@info.uqam.ca

Abstract. Autonomic networking is an emerging approach to the management of computer networks that aims at developing self-governed devices. Among the main issues of autonomic systems is the question of self-configuration. In this paper, we describe a method for discovering and self-generating the configuration of a network device in order to dynamically push a new service into a network. On each configuration, several rules representing the semantics of the services are expressed in a logical formalism called Configuration Logic. From these rules, we show how to use traditional satisfiability methods to automatically generate or modify the configuration of a device with respect to the configuration of its neighbours. We illustrate our case with an example of a switch that automatically discovers its VLAN configuration when connected to an existing network. The results presented here have been implemented into the configuration management tool ValidMaker.

1 Introduction

Despite the tremendous development of network services and functionalities over the years, the configuration and deployment of network elements such as routers and switches remains a mostly manual task. An intricate knowledge of each devices' and services' inner workings and dependencies between configuration parameters is required from the network engineer in order to successfully run even basic use cases. The addition of a new device or the deployment of a new service to an existing infrastructure requires repetitive but careful manipulation of multiple configuration parameters on many elements, and even such a cautious work can spawn unpredicted side effects that are discovered by trial and error.

The application of the autonomic systems paradigm [19] to computer networks offers a promising means to release the burden of knowledge and tedious manipulations currently needed from engineers. By definition, self-governed devices can automatically modulate their behaviour in reaction to configuration inconsistencies or changes in the topology or management policies of the network.

In this paper, we describe a method of automatically discovering and generating the configuration of a network device. Each configuration is represented in the form of a hierarchy of parameter-value pairs that is assimilated to a labelled

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 36–49, 2006.

tree. The dependencies between these parameters are then expressed as self-rules in a special formalism called Configuration Logic (CL) [22].

From the CL formula defining each rule, we show how an action can be automatically associated; this action consists in a number of configuration operations whose end result ensure that the rule is fulfilled. A CL validation engine can automatically check whether a given set of self-rules are respected in a network and trigger the appropriate actions associated with the rules that are violated.

The system then uses a standard Boolean satisfiability test to generate a plan that properly instantiates parameter values and chooses between conflicting actions; in the case where more than a single plan is possible, the final choice can then be passed on to a higher-level policy manager. This method can be used for integrating self-configuring and self-healing functionalities in any autonomic network where devices' configurations are represented by trees. We illustrate it by a simple example on Virtual LANs in which a new switch is connected and discovers its configuration in a plug-and-play manner based on data obtained from other devices in the network.

The rest of the paper is structured as follows. Section 2 presents related work. Section 3 presents the VLAN use case that illustrates our method and elicits a number of VLAN configuration self-rules. Section 4 presents the tree structure used to describe configurations and formalises the configuration rules using Configuration Logic. Section 5 extends the original VLAN scenario to allow for self-configuration and shows how violated VLAN rules trigger configuration operations. Finally, section 6 concludes and indicates possible future work.

2 Related Work

The autonomic approach to network management has been the source of numerous related work in the recent years. Companies like Cisco and Motorola are developing similar concepts respectively called *programmable networks* [2] and *cognitive networks*; the idea has also been developed in [16].

The SELFCON [5] project developed a self-configuring environment based on the Directory-enabled Networking (DEN) [21] principles, where devices register at a centralised directory server that notifies them of changes in configuration policies or network state. Similarly, the AUTONOMIA [15] environment provides an autonomic architecture for automated control and management of networked applications. In [20], a suite of protocols compatible with TCP/IP is described that could be implemented into autonomic, agent-based “smart routers” that take decisions about which protocol in the suite should be used to optimise some user-defined goals. All these projects describe an infrastructure in terms of high level concepts and do not concentrate on the representation, validation and actual generation of configurations and rules, but rather provide an environment in which these operations can take place. The agent approach is extended in [11] from a quality of service perspective and is currently under development.

In [12], the parameters of an optical network are automatically reconfigured based on link traffic using regression techniques. However, the range of legal values of these parameters is fixed and known in advance and the reconfiguration only aims at finding an optimal adjustment of existing values: the network itself is supposed to be properly working at any moment. Our work rather attempts to structurally modify a configuration by adding and removing parameters. Moreover, in our situation, the legal range of values changes from time to time and our method attempts to discover that range from the configuration rules.

The GulfStream software system [10] provides a dynamic topology discovery and failure detection for clusters of switches in multiple VLANs; the approach is not based on the examination of the configuration of other switches, but rather on the broadcasting of Beacon and heartbeat messages between VLAN peers and is somewhat restrained to this particular situation.

Our approach also differs from well-established configuration systems like Cfengine [7] in that only the desired properties of the configuration are expressed in a declarative way, but no action or script must be specified by the user. The method we present automatically determines the proper actions to take on the configuration in order to fulfill the desired rules.

Finally, a lot of work has been published about self-configuration applied to wireless sensor networks. In this context, the word "configuration" generally refers to the topological arrangment of the different elements forming the sensor mesh, and not the logical parameters that regulate the behaviour of a device in itself. It is therefore only faintly related to the present work.

3 Constraints and Rules in Network Services

In this section, we develop a configuration example based on Virtual Local Area Nertworks and the Virtual Trunking Protocol. We express configuration self-rules for a configuration of this type to be functional. This example will later be used to show how we can find the appropriate configuration of a device in a self-configuring and self-healing context.

3.1 Virtual Local Area Networks and the Virtual Trunking Protocol

Switches allow a network to be partitionned into logical segments through the use of Virtual Local Area Networks (VLAN). This segmentation is independent of the physical location of the users in the network. The ports of a switch that are assigned to the same VLAN are able to communicate at Layer 2 while ports not assigned to the same VLAN require Layer 3 communication. All the switches that need to share Layer 2 intra-VLAN communication need to be connected by a *trunk*. IEEE 802.1Q [4] and VTP [1] are two popular protocols for VLAN trunks.

In principle, for a VLAN to exist on a switch, it has to be manually created by the network enginneer on the said switch. The Virtual Trunking Protocol (VTP) [1] has been developped on Cisco devices to centralise the creation and deletion of VLANs in a network into a VTP *server*. This server takes care of creating, deleting, and updating the status of existing VLANs to the other switches sharing the same VTP *domain*.

3.2 Constraints on VLAN Configurations

Our configuration example involves VTP. Consider a network of switches such as the one shown in Figure 1 where several VLANs are available. We express a number of VTP self-rules that must be true across this network.

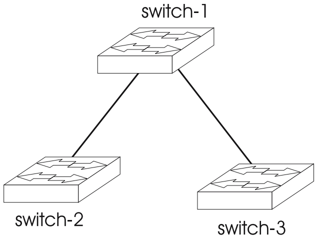


Fig. 1. A simple cluster of switches in the same VLAN. The links are VLAN trunks.

First, in order to have a working VTP configuration, the network needs a unique VTP server; all other switches must be VTP clients. This calls for a first set of two self-rules:

VTP Self-Rule 1. *The VTP must be activated on all switches.*

VTP Self-Rule 2. *There is a unique VTP server.*

We impose that all switches be in the same VTP domain, and that if two switches are connected by a trunk, then this trunk must be encapsulated in the same mode on both interfaces. This gives us two more constraints that should be true in all times:

VTP Self-Rule 3. *All switches must be in the same VTP domain.*

VTP Self-Rule 4. *The interfaces at both ends of a trunk should be defined as such and encapsulated in the same mode.*

4 Configurations and Rules

In this section, we give a brief overview of Configuration Logic (CL), a logical formalism over tree structures developed for expressing properties of configurations. We show how the VTP Self-Rules described in the previous section can be rewritten in CL to become Formal VTP Self-Rules.

4.1 Representing Configurations

As explained in [13], the configuration of network devices such as routers and switches can be represented as a tree where each node is a pair composed of a name and a value. This tree represents the hierarchy of parameters inherent to

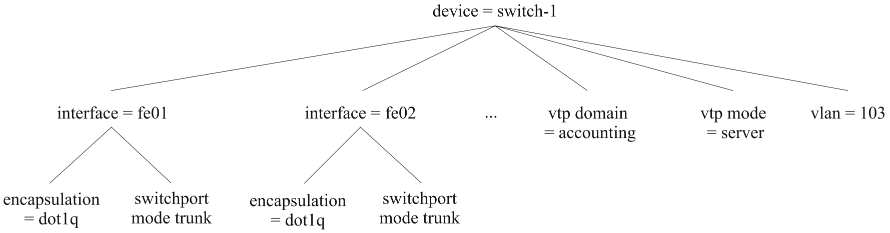


Fig. 2. A portion of the configuration of the `switch-1` in the network of Figure 1. The configuration of `switch-2` and `switch-3` differs in the VTP mode and trunk information.

the configuration of such devices. As an example, Figure 2 shows a tree representation of the configuration of `switch-1` in the network of Figure 1.

Building a tree from an XML document is trivial; therefore, the representation of configurations as trees closely matches the XML nature of a protocol such as Netconf [9] that uses this format to fetch and modify the configuration of a device.

4.2 A Logic for Configurations

Configuration Logic [22] was developped in order to express properties on configuration trees. CL formulas use the traditional Boolean connectives of predicate logic: $\wedge$ ("and"), $\vee$ ("or"), $\neg$ ("not"), $\rightarrow$ ("implies"), to which two special quantifiers are added. The universal quantifier, identified by [ ], indicates a path in the tree and imposes that a formula be true for all nodes at the end of that path. Likewise, the existential quantifier, identified by $\langle\ \rangle$, indicates a path in the tree and imposes that a formula be true for some node at the end of that path.

To improve readability of CL rules, we can use predicates. Predicates are expressed in the same way as rules, with the exception that arguments which correspond to already quantified nodes can be specified. For example, to create a predicate that returns true if a switch is a VTP Client, one can do so by writing:

$$\begin{array}{l}\text{IsVTPClient}(S)\ :-\\ \quad \langle S\,;\,\text{vtp mode} = x\rangle\, x = \text{client}\end{array}$$

This predicate states that under the node S passed as an argument, there exists a node whose name is "vtp mode" and whose value is x, and where x is equal to "client". In other words, this predicate is true whenever S has a child labelled "vtp mode = client", which means that the device is indeed a VTP client.

Similarly, the predicates IsVTPServer and IsTrunk respectively indicate that the device is a VTP server and that a specific interface is connected to a trunk. They are defined as the following:

$$\begin{array}{l}\text{IsVTPServer}(S)\ :-\\ \quad \langle S\,;\,\text{vtp mode} = x\rangle\, x = \text{server}\end{array}$$

$$\begin{array}{l}\text{IsTrunk}(I)\ :-\\ \quad \langle I\,;\,\text{switchport mode} = x\rangle\, x = \text{trunk}\end{array}$$

The next predicate asserts that two switches are in the same VTP domain. This is done by checking that for two nodes S and T representing the root of the configuration tree of two devices, every VTP domain listed under S also appears under T.

$$\begin{array}{l}\text{SwitchesInSameVTPDomain}(S, T)\ :-\\ \quad [S\,;\,\text{vtp domain} = x]\\ \qquad \langle T\,;\,\text{vtp domain} = y\rangle\ x = y\end{array}$$

Finally, this last predicate verifies that the encapsulation on a VLAN trunk is either IEEE 802.11Q or ISL, and that both ends use matching protocols:

$$\begin{array}{l}\text{SameEncapsulation}(I_1, I_2)\ :-\\ \quad [I_1\,;\,\text{switchport encapsulation} = x_1]\\ \qquad \langle I_2\,;\,\text{switchport encapsulation} = x_2\rangle\\ \qquad\quad (x_1 = \text{dot1q} \wedge x_2 = \text{dot1q}) \vee (x_1 = \text{isl} \wedge x_2 = \text{isl})\end{array}$$

For more details about CL, the reader is directed to [22,23,14].

4.3 Rules Expressed in Configuration Logic

Equipped with the previous predicates, the VTP self-rules mentioned in section 3 can now be expressed in Configuration Logic. The first rule checks whether VTP is activated on all switches.

Formal VTP Self-Rule 1 (VTPActivated)

$$\begin{array}{l}[\text{device} = s_1]\\ \quad \text{IsVTPServer}(s_1) \vee \text{ISVTPClient}(s_1)\end{array}$$

This is done by stating that for every device s_1, either s_1 is a VTP server, or s_1 is a VTP client. Note that this rule uses the predicates defined above to simplify its expression; however, the predicates are not necessary, and their definition could have simply been copied in the rule instead.

The second rule makes sure that there is one, and only one server in the network. It first states that there exists a device s_1 which is a VTP server, and then that every device s_2 different from s_1 is a VTP client. Remark that this rule subsumes the previous one, which has still been included to illustrate later concepts.

Formal VTP Self-Rule 2 (UniqueServer)

$$\begin{array}{l}\langle\text{device} = s_1\rangle\\ \quad (\text{IsVTPServer}(s_1) \wedge\\ \qquad [\text{device} = s_2]\\ \qquad\quad s_1 \neq s_2 \rightarrow \text{IsVTPClient}(s_2))\end{array}$$

The third rule checks that all switches in the network have the same VTP Domain.

Formal VTP Self-Rule 3 (SameVTPDomain)

$$\begin{array}{l}
[\text{device} = s_1] \\
\quad [\text{device} = s_2] \\
\quad\quad \text{SwitchesInSameVTPDomain}(s_1, s_2)
\end{array}$$

It does so by stating that for every pair of devices s_1 and s_2, the predicate "SwitchesInSameVTPDomain" defined previously is true.

Finally, the fourth rule checks that if two interfaces are connected, then these two interfaces must be defined as trunks and have the same encapsulation protocol.

Formal VTP Self-Rule 4 (TrunkActive)

$$\begin{array}{l}
[\text{device} = s_1] \\
\quad [\text{device} = s_2] \\
\quad\quad [s_1\,;\, \text{interface} = i_1] \\
\quad\quad\quad \langle s_2\,;\, \text{interface} = i_2 \rangle \\
\quad\quad\quad\quad (\text{InterfacesConnected}(i_1, i_2) \;\rightarrow (\text{IsTrunk}(i_1) \\
\quad\quad\quad\quad\quad \wedge\, \text{IsTrunk}(i_2) \;\wedge\, \text{SameEncapsulation}(i_1, i_2)))
\end{array}$$

The relation "InterfacesConnected" is not a predicate defined in CL, but rather a system primitive that the network can tell us about. It returns true if the two interfaces are connected by a link. This requires knowledge of the topology of the network and depends on the architecture of the autonomic component, as will be discussed in the next section.

5 A Self-configuration Scenario

In this section, we show how to use Configuration Logic to trigger configuration changes in devices. Figure 3 presents a possible architecture using this method. It shows how an existing tool called ValidMaker [8] that downloads and uploads configuration trees from the configuration file stored on routers and switches, can be used off-the-shelf to provide autonomic behaviour to the devices. A CL validation engine is implemented within ValidMaker. Therefore, one can load configurations, define self-rules on these configurations and automatically verify them. If one or many rules happen to be false, an additional configuration generator can adjust the configuration trees and generate the new configurations which can then be put back on the devices.

To this end, we revisit the original VLAN scenario described in section 3 from a network autonomic viewpoint. Instead of validating configuration rules on a static, fully formed cluster of switches, we suppose one switch is connected to the cluster with a blank configuration, as shown in Figure 4. More precisely, the interface fe04 of `switch-4` is connected to the interface fe06 of `switch-3`. The initial configuration tree of `switch-4` is is a tree with only one root node labelled "device=switch-4".

Based on this use case, we describe a method that enables the switch to automatically discover its configuration.

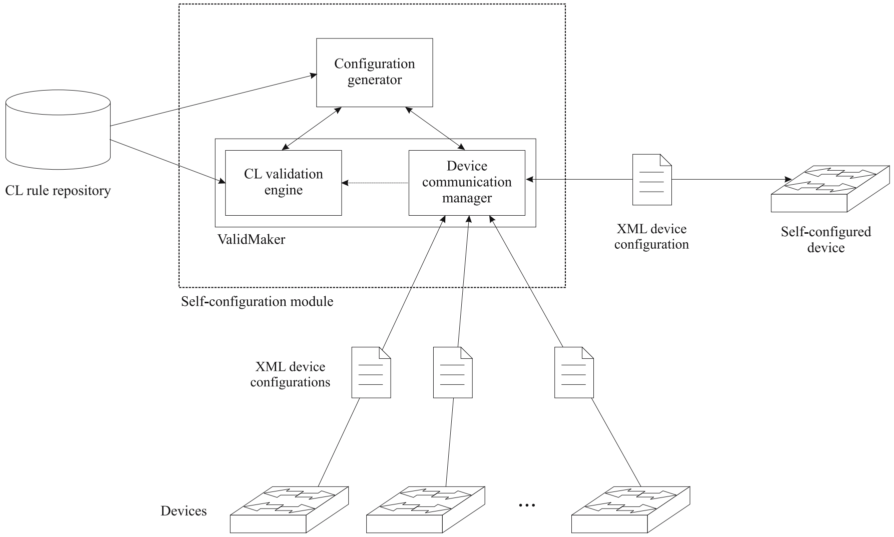


Fig. 3. A possible self-configuring architecture using off-the-shelf components

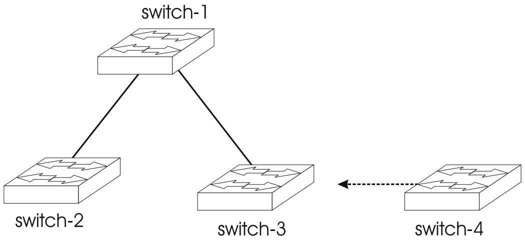


Fig. 4. Autonomic use case. Switch 4 is connected to Switch 3 and attempts to discover its configuration.

5.1 Active Constraints

The network rules shown in section 3.2 have been up to now interpreted in a static way: their validation allowed us to determine whether a configuration satisfied them or not. However, it is possible to go further and automatically associate with each rule an action, thereby making it a “self-rule”. This action, in line with the semantics of the operators used in the rule, is generated such that executing it changes the configuration in a way that fulfills the rule that was originally proved false.

As an example, consider a rule of the form $\langle \mathrm{p} = x \rangle\, \varphi(x)$. Such a rule imposes the existence of a node $\mathrm{p} = x$ in the configuration of the device. If it is violated, this means that no such tuple $\mathrm{p} = x$ exists; to make this rule true, the node must be added to the configuration. This addition of a new node corresponds to a configuration operation equivalent to typing a command (or a sequence of commands) to the CLI of the device. Moreover, the value x of this parameter must be such that $\varphi(x)$ is true.

For such a procedure to be possible, the formulas it attempts to fulfill must be expressed in a logic where the existence of a model is a *decidable* problem. Otherwise, it is not always possible to find and construct a configuration satisfying a given formula. Therefore, the choice of CL as the logical formalism of formal rules is not arbitrary: it has been shown in [23] that under reasonable conditions, CL is a decidable logic. Using a richer logic such as XPath or TQL would lead to undecidability, as has been shown in [6].

Hence, by recursively descending through the nested CL subformulas, it is possible to come up with a series of configuration operations and constraints on their parameters that make every self-rule true. To this end, we define a procedure called $\textsc{Plan}_T$ that recursively creates the plans for each rule given a global configuration tree T. The special notation $\oplus(\overline{p} = \overline{x}\,; p = x)$ indicates that a node of parameter p and value x must be added to the tree at the tip of the branch $\overline{p} = \overline{x}$. This procedure is defined in Table 1.

Table 1. The recursive plan generation procedure

$$\begin{aligned}
\textsc{Plan}_T(\varphi) &= 1 \text{ if } T \models \varphi \\
\textsc{Plan}_T(\neg\varphi) &= \neg\textsc{Plan}_T(\varphi) \\
\textsc{Plan}_T(\varphi \wedge \psi) &= \textsc{Plan}_T(\varphi) \wedge \textsc{Plan}_T(\psi) \\
\textsc{Plan}_T(\varphi \vee \psi) &= \textsc{Plan}_T(\varphi) \vee \textsc{Plan}_T(\psi) \\
\textsc{Plan}_T(\langle \overline{p} = \overline{x}\,; p = x\rangle\ \varphi(x)) &= \oplus(\overline{p} = \overline{x}\,; p = x) \wedge \textsc{Plan}_T(\varphi) \\
\textsc{Plan}_T([\overline{p} = \overline{x}\,; p = x]\ \varphi(x)) &= \bigwedge_x \textsc{Plan}_T(\varphi)
\end{aligned}$$

As an example, consider the initial configuration of `switch-4` when added to the configuration of the original cluster of switches shown in Figure 2. It is clear that Formal VTP Self-Rule 1, which imposes the presence of either server or client definition on every switch, is violated for `switch-4`. The recursive application of $\textsc{Plan}$ on this formula produces the following action:

$$\begin{gathered}
\oplus(\text{device} = \text{switch-4}\,; \text{vtp mode} = \text{server}) \vee \\
\oplus\ (\text{device} = \text{switch-4}\,; \text{vtp mode} = \text{client})
\end{gathered} \tag{1}$$

which commands that the VTP role of the switch be determined. This role is specified by the addition of either node vtp mode = server or vtp mode = client under the root of `switch-4`'s configuration tree.

Similarly, Formal VTP Self-Rule 2 is violated. It imposes that there is only one server in the network and that all other be clients, and produces the following action:

$$\oplus\ (\text{device} = \text{switch-4}\,; \text{vtp mode} = \text{client}) \tag{2}$$

In other words, the only possible action is the definition of `switch-4` as a client.

Formal VTP Self-Rule 3 is also violated. This rule imposes that all switches be in the same domain. The application of Plan on this rule produces this action:

$$\oplus(\text{device} = \text{switch-4}\,;\ \text{vtp domain} = \text{accounting}) \tag{3}$$

Finally, the actions produced by Formal VTP Self-Rule 4 are the following:

$$\begin{array}{l}
\oplus(\text{device} = \text{switch-3}\,;\ \text{interface} = \text{fe06})\\
\ \wedge \oplus(\text{device} = \text{switch-4}\,;\ \text{interface} = \text{fe04})\\
\ \wedge \oplus(\text{device} = \text{switch-3}\,,\ \text{interface} = \text{fe06}\,;\ \text{switchport mode} = \text{trunk})\\
\ \wedge \oplus(\text{device} = \text{switch-4}\,,\ \text{interface} = \text{fe04}\,;\ \text{switchport mode} = \text{trunk})\\
\ \wedge (\\
(\oplus(\text{device} = \text{switch-3}\,,\ \text{interface} = \text{fe06}\,;\ \text{switchport encapsulation} = \text{dot1q}) \wedge\\
\ \oplus\ |!(\text{device}{=}\text{switch-4}\,,\ \text{interface}{=}\text{fe04}\,;\ \text{switchport encapsulation}{=}\text{dot1q})) \vee\\
(\oplus(\text{device} = \text{switch-3}\,,\ \text{interface} = \text{fe06}\,;\ \text{switchport encapsulation} = \text{isl}) \wedge\\
\ \oplus\ (\text{device} = \text{switch-4}\,,\ \text{interface} = \text{fe04}\,;\ \text{switchport encapsulation} = \text{isl}))\\
)
\end{array} \tag{4}$$

This last set of actions is interesting, since it imposes addition of nodes not only under the newly connected `switch-4`, but also under `switch-3` which is already part of the working VLAN. The reason is that connecting `switch-3` to `switch-4` involves setting up a trunk, an operation that requires configuration modifications on both ends of the wire.

Therefore, the actual course of actions will depend on the architecture chosen: if the self-configuration module depicted in Figure 3 is decentralised in every switch, then each switch must ignore the actions that modify other devices, assuming that the concerned devices will take proper measures to correct the situation on their side. If the self-configuration module is rather centralised for some number of devices, the changes can be sent by the device communication manager to multiple switches at once.

The global plan that must be executed to make the rules true is the conjunection of equations (1)-(4).

5.2 Generating the Configuration

It can be seen that the previous plan is nondeterministic. For example, if a formula $\langle \mathrm{p} = x \rangle\, \varphi \vee \psi$ is false, there are two different ways of making it true: either by making φ true, or by making ψ true. In such a case, both solutions are explored and propagated up the recursion stack. This happens with the actions produced by the Formal VTP Self-Rule 1 that either define the switch as client or as server. In the same way, the actions produced by the Formal VTP Self-Rule 4 rule impose that the trunk encapsulation in `switch-3` and `switch-4` be either both dot1q or both isl, but since none of them is already configured and no further constraint applies, the choice is free.

Moreover, a separate plan has been generated for each violated self-rule; however, it is well possible that these plans are mutually contradictory and that no global solution exists. Finally, one must make sure that executing the plans not only makes the violated rules true, but in turn does not produce side effects that could falsify other rules that were previously true.

To this end, two further steps are taken before committing any configuration to the device. First, a satisfying assignment of the global plan must be found. This can be easily obtained by applying a standard Boolean satisfiability (SAT) solver such as zChaff [17] or SATO [24]. This assignment designates the actions that are actually chosen from the many alternatives suggested by the plan to make the formulas true.

Once this assignment has been found, the plan is tentatively executed on the configuration, and the whole set of self-rules is then revalidated against this modified configuration. If one of the rules is violated, the satisfying assignment is discarded and the procedure backtracks to find a new assignment. This is the case if the following operations are chosen as the action to execute on the configuration (for clarity, the actions on `switch-3` have been omitted):

$$\begin{gathered}
\oplus(\text{device} = \text{switch-4}\,;\ \text{vtp mode} = \text{server})\\
\oplus(\text{device} = \text{switch-4}\,;\ \text{vtp mode} = \text{client})\\
\oplus(\text{device} = \text{switch-4}\,;\ \text{vtp domain} = \text{accounting})\\
\oplus(\text{device} = \text{switch-4}\,;\ \text{interface} = \text{fe04})\\
\oplus(\text{device} = \text{switch-4}\,,\ \text{interface} = \text{fe04}\,;\ \text{switchport mode} = \text{trunk})\\
\oplus(\text{device} = \text{switch-4}\,,\ \text{interface} = \text{fe04}\,;\ \text{switchport encapsulation} = \text{dot1q})
\end{gathered}$$

The configuration resulting of the previous operations violates Formal VTP Self-Rule 2, since it assigns `switch-4` both to the role of client and server at the same time. It violates Formal VTP Self-Rule 2 for another reason, as a server already exists in the network.

However, a second possible truth assignment to the global plan is the following:

$$\begin{gathered}
\oplus(\text{device} = \text{switch-4}\,;\ \text{vtp mode} = \text{client})\\
\oplus(\text{device} = \text{switch-4}\,;\ \text{vtp domain} = \text{accounting})\\
\oplus(\text{device} = \text{switch-4}\,;\ \text{interface} = \text{fe04})\\
\oplus(\text{device} = \text{switch-4}\,,\ \text{interface} = \text{fe04}\,;\ \text{switchport mode} = \text{trunk})\\
\oplus(\text{device} = \text{switch-4}\,,\ \text{interface} = \text{fe04}\,;\ \text{switchport encapsulation} = \text{dot1q})
\end{gathered}$$

The configuration resulting of these operations validates all formal VTP self-rules and is therefore admissible. In such a case, the tentative configuration is committed as the running configuration of the switch. The resulting configuration tree obtained from these autonomic configuration steps is shown in Figure 5.

One can remark that this method can also work in a self-healing context where a working network is disturbed; the recursive plan generation method provided here ensures that the configurations of the devices can be but back to a state where they fulfill all the rules that must apply on the network.

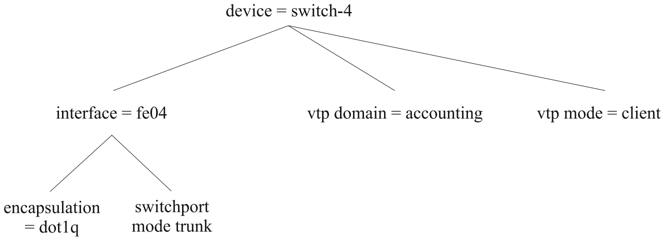


Fig. 5. The configuration of `switch-4` after automatic configuration generation

Table 2. Validation time of each formal VTP self-rule in a network formed of a varying number of switches

	Validation time per device (ms)			
Switches	Rule 1	Rule 2	Rule 3	Rule 4
10	0.024	0.012	0.106	0.564
20	0.029	0.012	0.312	1.715
40	0.044	0.024	1.011	6.482
80	0.078	0.041	3.655	21.948

5.3 Complexity of the Method and Experimental Results

We now evaluate the global complexity of the method. Although Boolean satisfiability is an NP-complete problem, it is not the bottleneck of the procedure, as the global Boolean formula that is generated never has more than a few dozens, possibly a hundred, configuration operations. Therefore, the size of the Boolean instance to be processed is negligible by SAT standards, and can be processed in fractions of a second by SAT solvers accustomed to problems topping the million of variables.

Each configuration operation in itself consists solely in the addition of a single node in the configuration tree of a device and can also be considered instantaneous. Hence, the core of the complexity of this method resides in the validation of CL formulas on multiple incarnations of configurations for many devices.

In order for this method to be tractable, the validation of CL formulas must be quick and simple. It has been shown in [14] that CL model checking of a formula is polynomial in the size of the tree.

We give in Table 2 the validation times for a network composed of 10 to 80 switches. These configurations were generated by a parameterisable script and then loaded into ValidMaker. All results have been obtained on a Pentium IV of 2.4 GHz with 512 Mb of RAM running Windows XP Professional; they are consistent with the polynomiality result already demonstrated.

As one can see from these results, the validation times are reasonable and do not excess 25 milliseconds per device for the largest network of 80 switches. One should compare these figures with the time required for actually injecting a new configuration in a switch and restarting it to make it effective, which is at least a few seconds. Moreover, it should be noted that the validation experiments shown here involve the validation of the whole network, and not only of the few devices that might be concerned when a new switch is connected; this is especially true for VTP Self-Rule 4.

6 Conclusion and Future Work

In this paper, we have shown how to self-configure a network device by validating constraints on the configuration of other devices expressed as trees of parameters and values. We showed how a new switch connected to an VLAN can discover its configuration by selecting a suitable plan that tries to fulfill the existing configuration constraints. We also showed by experimental results with the ValidMaker configuration tool that the validation of self-rules is a tractable operation that imposes negligible time to execute.

At the moment, the SAT instances forming the possible plans are treated separately from the CL validation in an independent solver embedded into the Configuration generator. A natural extension of this work is to enrich this active model by dealing with side effects of the application of a self-rule and support node deletion and modification, for example by integrating CL components into an existing SAT solver in the context of SAT modulo theories [18].

Acknowledgement

We gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada on this research.

References

1. Configuring VTP. http://www.cisco.com/en/US/products/hw/switches/ps708/products_configuration_guide_chapter09186a008019f048.html.
2. An OSS for packet networks. *Cisco Systems Packet Magazine*, 14(1):25–27, January 2002.
3. 802.11Q: Virtual bridged local area networks standard, 2003. http://standards.ieee.org/getieee802/download/802.1Q-2003.pdf.
4. 802.11Q: Virtual bridged local area networks standard, 2003. http://standards.ieee.org/getieee802/download/802.1Q-2003.pdf.
5. R. Boutaba, S. Omari, and A. P. S. Virk. SELFCON: An architecture for self-configuration of networks. *Journal of Communications and Networks*, 3(4): 317–323, December 2001.
6. C. Calcagno, L. Cardelli, and A. D. Gordon. Deciding validity in a spatial logic for trees. In *TLDI*, pages 62–73, 2003.
7. A. L. Couch and M. Gilfix. It's elementary, dear Watson: Applying logic programming to convergent system management processes. In *LISA*, pages 123–138. USENIX, 1999.
8. R. Deca, O. Cherkaoui, and D. Puche. A validation solution for network configuration. In *CNSR*, 2004.

9. R. Enns. Netconf configuration protocol, IETF internet draft, February 2006.
10. S. A. Fakhouri, G. S. Goldszmidt, M. H. Kalantar, J. A. Pershing, and I. Gupta. Gulfstream - a system for dynamic topology management in multi-domain server farms. In *CLUSTER*, pages 55–62. IEEE Computer Society, 2001.
11. D. Gaïti, G. Pujolle, M. Salaün, and H. Zimmermann. Autonomous network equipments. In I. Stavrakakis and M. Smirnov, editors, *WAC*, volume 3854 of *Lecture Notes in Computer Science*, pages 177–185. Springer, 2005.
12. W. Golab and R. Boutaba. Optical network reconfiguration using automated regression-based parameter value selection. In *ICN*, 2004.
13. S. Hallé, R. Deca, O. Cherkaoui, and R. Villemaire. Automated validation of service configuration on network devices. In J. B. Vicente and D. Hutchison, editors, *MMNS*, volume 3271 of *Lecture Notes in Computer Science*, pages 176–188. Springer, 2004.
14. S. Hallé, R. Villemaire, and O. Cherkaoui. CTL model checking for labelled tree queries. In *TIME*, pages 27–35. IEEE Computer Society, 2006.
15. S. Hariri, L. Xue, H. Chen, M. Zhang, S. Pavuluri, and S. Rao. Autonomia: An autonomic computing environment. In *IPCCC*, April 2003.
16. R. M. Keller. *Self-Configuring Services for Extensible Networks – A Routing-Integrated Approach.* PhD thesis, Swiss Federal Institute of Technology, 2004.
17. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering and efficient SAT solver. In *Proceedings of the 39th Design Automation Conference*, June 2001.
18. R. Nieuwenhuis and A. Oliveras. Decision procedures for SAT, SAT modulo theories and beyond. the BarcelogicTools. In G. Sutcliffe and A. Voronkov, editors, *LPAR*, volume 3835 of *Lecture Notes in Computer Science*, pages 23–46. Springer, 2005.
19. M. Parashar and S. Hariri. Autonomic computing: An overview. In J.-P. Banâtre, P. Fradet, J.-L. Giavitto, and O. Michel, editors, *UPP*, volume 3566 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2004.
20. G. Pujolle and D. Gaïti. Intelligent routers and smart protocols. In *INTELLCOMM*, pages 16–27, 2004.
21. J. Strassner. *Directory Enabled Networks.* Macmillan Technical Publishing, 1999.
22. R. Villemaire, S. Hallé, and O. Cherkaoui. Configuration logic: A multi-site modal logic. In *TIME*, pages 131–137. IEEE Computer Society, 2005.
23. R. Villemaire, S. Hallé, and O. Cherkaoui. A hierarchical logic for network configuration. In *LPAR*, 2005. Short paper proceedings.
24. H. Zhang and M. E. Stickel. Implementing the Davis-Putnam method. *J. Autom. Reasoning*, 24(1/2):277–296, 2000.

Dynamic Decision Making for Candidate Access Point Selection

Burak Simsek[1], Katinka Wolter[1,*], and Hakan Coskun[2]

[1] Institut für Informatik, HU Berlin
Unter den Linden 6, 10099 Berlin
simsek@informatik.hu-berlin.de, wolter@informatik.hu-berlin.de
[2] ETS, TU Berlin Franklinstr.28/29
10587 Berlin
coskun@cs.tu-berlin.de

Abstract. In this paper, we solve the problem of candidate access point selection in 802.11 networks, when there is more than one access point available to a station. We use the QBSS (quality of service enabled basic service set) Load Element of the new WLAN standard 802.11e as prior information and deploy a decision making algorithm based on reinforcement learning. We show that using reinforcement learning, wireless devices can reach more efficient decisions compared to static methods of decision making which opens the way to a more autonomic communication environment. We also present how the reinforcement learning algorithm reacts to changing situations enabling self adaptation.

1 Introduction

The existence of more than one access point available to one station is usual for many daily applications of WLAN. In such cases, the station should be able to decide on one of the access points optimally. At this point, some kind of information that is helpful for being able to estimate to what extent the candidate access points can satisfy the required service quality is essential. This requires the knowledge of the most informative network parameters among many possible. Currently, the QBSS (quality of service enabled basic service set) Load Element of the WLAN standard 802.11e [8] seems to be the most appropriate information element that could be used to choose the access points which can offer the required QoS (quality of service). It is responsible for informing the stations about the load of an access point so that stations can decide on associating themselves with the access point or not. Including three parameters, this element is periodically sent to all of the stations within the beacon frames. These parameters are station count, available admission capacity and channel utilization.

However, in our previous study we showed by correlation analysis that the parameters of the QBSS load element are not reliable in most of the cases [14].

[*] The authors Burak Simsek and Katinka Wolter would like to thank German Research Foundation (DFG) for support of this work under grant number WO 898/1-2.

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 50–63, 2006.

This is due to the fact that there is no high correlation with the QBSS load element parameters and the QoS metrics such as delay, jitter and loss. Unfortunately, choosing an access point with less channel utilization or less stations associated with the access point is the most used approach which does not lead to efficient decision making in 802.11e networks.

In this paper we find out the applicability, reliability and performance of a more sophisticated decision making algorithm for the candidate access point selection problem which is based on reinforcement learning. We show that the reinforcement learning algorithm (RLA) enables the stations to reach higher decision accuracy compared with the traditional methods. We deploy regression analysis for performance comparison purposes since regression analysis includes but not restricted to the traditional logic such as "choose the access point with less station count or channel utilization". Unfortunately there is no other study known to the authors which deals with the use of intelligent decision making methods for the candidate access point selection problem. We demonstrate that the algorithm does not pose additional load to the system during runtime if it is trained prior to the deployment by the end customers of wireless devices implementing 802.11e. Following this, we propose an approach to engage RLA over wireless devices in an efficient manner. We show how a trained algorithm can adapt itself to dynamic environments which introduces a significant advantage when compared with the statistical way of decision making and therefor enables an autonomic communication environment.

Since the 802.11e standard is new, most of the relevant studies made so far include performance analysis and improvement [3]. To the best of our knowledge, there is no study working on the candidate access point selection problem and the evaluation of the QBSS load element in solving this problem. Additionally, although there are services enabling intelligent decision making over 802.11e networks, there is no study known to us in this direction. Reinforcement learning is a well known algorithm of artificial intelligence which has been used very often in different fields of scientific research. Soccer robots [5] and chess and backgammon tools [6] are some examples. The main reason why we applied reinforcement learning to the candidate access point selection problem is that it is relatively easier to deploy over wireless devices. It is less complex when compared with other learning algorithms of artificial intelligence. [11] gives an enhanced study of such algorithms. However despite their complexities and problems in solving infinite size problems [15], simplified versions of evolutionary algorithms [9] might also prove to be effective which is another research problem. The main merit of this study is that it prepares the basics for a network management methodology by the means of a dynamic decision making algorithm over 802.11e. We also deploy the same algorithm for call admission control using the traffic specification element of the protocol IEEE 802.11e. Due to size restrictions, the results of admission control with reinforcement learning are going to be presented within our next study.

The rest of the paper is structured as follows. In the second section, we introduce the candidate access point selection problem that we want to solve and summarize the decision making algorithms. In the third section we describe the simulation environment. The results of the simulation are given in the fourth section. In the fifth section we present how RLA can adapt to dynamic environments. The sixth section makes the conclusion by comparing the results and consequently making recommendations.

2 Decision Making

2.1 Regression Analysis

The main goal of regression analysis is to estimate the effect of a change in one of the independent variables of a system on a dependent variable within the system by using some observations of the dependent and the independent variables. One of the simplest forms of regression analysis is the linear regression as given in equation (1).

$$y_i = a_0 + a_1 x_{i,1} + a_2 x_{i,2} + ... + a_n x_{i,n} \text{ i < m} \tag{1}$$

where n is the number of independent variables, m the number of dependent variables, y_i the dependent variables, $x_{i,k}$ the independent variables and a_k the parameters to be estimated. During our study, we tried a number of possible functions like linear, log-linear or non-linear functions in order to estimate the expected QoS using the QBSS load element parameters. In this paper we selected to present only the results of the linear function because despite its simplicity, the amount of correct decisions was the highest in most of the cases. A typical equation for QBSS load elements is given as follows:

$$E[QoS] = a * StaCount + b * ChanUtil + c * AdCap + d \tag{2}$$

where StaCount is the number of stations associated with an access point, ChanUtil is the percentage of time that the channel being used by the access point is sensed to be busy, AdCap is the remaining amount of time in the HCCA scheduler and d is the disturbance term. We try to estimate a,b,c and d for this function and use simulation results as the input data for this purpose. Although the station count and channel utilization appear to have a reverse ratio to the expected QoS and available admission capacity a direct ratio, we showed in [14] that this assumption is not true in many cases for 802.11e. Additionally during regression analysis, the parameters of the variables having a reverse ratio to the dependent variable are negative. Because of these two reasons, it makes sense to have the variables, which we expect to have a revere ratio with QoS, as a nominator within the equation (2).

2.2 Reinforcement Learning

A formal definition of a reinforcement learning problem in our case can be written as follows: An RL decision making algorithm receiving the QBSS load element information is at state s_t(x, y, z | x ϵ $\mathbb{Z}$, y ϵ 0..255, z ϵ 0..65536) where x is the total station count, y is the normalized channel utilization of the access point and z is the amount of medium time available in units of 32 microseconds, and has to make an action a_t(associate, do not associate). Each action a_t brings reward R_t to the algorithm. The goal of the RL algorithm is to select actions which maximize the sum of its discounted rewards R_t, where the discount factor is γ ϵ [0, 1]. In order to do this, RL algorithm has to decide about the value of an action at each state which is given as Q(s_t,a_t). The optimal Q(s_t,a_t) satisfies the following equation:

$$Q(s_t, a_t) = \left(R_t + \gamma \sum_{j=S} P_{s,j}\,(a_t)V(j) \right) \tag{3}$$

Here, V(j) is the maximum Q value that is expected to occur after a transition from s_t to j with respect to all actions a_t. $P_{s,j}$(a_t) is the transition probability from state s_t to state j. For further information about our RL algorithm and the choice of the parameters please see [13].

2.3 Problem Definition

We define two dynamic decision making problems.

1. Given an access point at state s_t, the decision function $f(s_t)$ which might be the function of the regression analysis or the RLA, a desired QoS level QoS^* and the real QoS after simulations QoS,
   - if $f(s_t)$ >= QoS^* then decision = accept else decision = not accept.
   - if decision = accept and QoS >= QoS^* then decision is accurate else decision is false.

   This problem is equivalent to estimating if an access point is going to be able to provide a service with sufficient QoS.

2. Given two access points at states $s_{1,t}$ and $s_{2,t}$
   - if $f(s_{1,t})$ >= $f(s_{2,t})$ then decision = 1 else decision = 2.
   - if QoS_1 >= QoS_2 then decision is accurate else decision is false

   which is equivalent to finding the better access point. The problem is not different when there are more access points. In such cases expected QoS for each access point should be calculated independently and then compared with each other.

We use the percentage of accurate decisions made by regression analysis and the RLA in order to decide on:

- to what extent the QBSS load element can help in choosing the right access point,
- which one of the decision making algorithms (reinforcement learning and regression analysis) help making more accurate decisions and in which instances.

Although a decision making method is successful in one of these problems, this might not be the case for the other problem. For example, even though a decision making algorithm can estimate which access point is better than the other, this does not guarantee that it can also estimate the real level of QoS. This is also true vice versa. Hence we have to differentiate between those two problems throughout the paper.

3 Simulation

3.1 Simulation Environment

The simulated network consists of one or two QoS enabled access points (QAP) depending on which one of the above defined problems is being solved and a random number of stations associated with the QAPs. Each station sends one type of traffic stream from the the list given below where up to 7 voice, 3 video, 10 interactive and 10 background streams are allowed to associate with an access point. Network is simulated using a slightly modified version of the 802.11e ns2 model of Ni [4].

In [12] it was shown that the current public hotspots are mainly occupied with low load TCP flows and only a few of the connected users demand real time applications based on high load UDP traffic. Additionally, the percentage of real time services is going to increase as the number of applications in this area increases [7]. Hence the network environment is designed to include 4 different traffic types. The first type is defined for voice traffic with the codec G.711. The second priority is defined for video traffic with different qualities which comprises most of the video codecs being used in the internet [10]. Third and fourth traffic types are defined to simulate normal hot spot user behavior as given in [12]. These traffic types are defined as follows:

1. Bidirectional constant bit rate (CBR) voice traffic using UDP with a packet size of 160 bytes and packet interval 20ms (8 Kbytes/s) corresponding to the VoIP codec G.711. (1st access category)
2. 12 simulated VBR video traffic streams using UDP with minimum packet size of 28 and maximum packet size of 1024 bytes with an average packet interval of 23ms corresponding to 30Kbytes/s. (2nd access category) (Average Quality Video)
3. Bidirectional interactive traffic using TCP with a packet size of 1100 bytes and exponentially distributed arrival rates having an average of 50ms on time, 30ms off time and sending rate of 60Kbits/s during on times corresponding to an average of 10Kbytes/s. This complies with the interactive traffic definition of 3GPP TS 22.105 and ITU G.1010. (3rd access category)
4. VBR Background traffic using TCP with a packet size of 1200 bytes and exponentially distributed inter arrival times having an average of 1000ms off and 200ms on times with a sending rate of 100Kbits/s corresponding to low load 11Kbytes/s traffic. (4th access category) (3GPP TS 22.105 Web Browsing- HTML definition.)

Table 1. List of Simulation Parameters

Bandwidth	11Mbps
PLCPTransmissionrate	1 Mbps
RTSThreshold	3000μs
ShortRetryLimit	7
LongRetryLimit	4
slotTime	9μs
AIFS(1,2,3,4)	$1, 2, 6, 12$
CWmin(1,2,3,4)	$7, 15, 15, 31$
CWMax(1,2,3,4)	$15, 31, 255, 525$

Additionally the 802.11e specific parameters are given in table 1.

The type of the traffic we are interested in throughout the paper is the voice traffic. As a metric of QoS for voice traffic we use the mean opinion score (MOS) values defined in ITU-T Rec. G.107 which is the widely accepted metric of industrial organizations to measure the quality of VoIP applications [2,1]. MOS rates voice calls on a scale of 1 to 5. It shows the satisfaction of real people who use a connection and rate the quality of voice signal after it has passed through a network from a source (transmitter) to a destination (receiver).

The most dominant enhancement in 802.11e MAC is the introduction of two new functions, the enhanced distributed channel access (EDCA) for differentiated services and the HCF controlled channel access (HCCA) for integrated services. Although HCCA and EDCA are compulsory within the standard, the amount of time reserved for these functions can be varied. In [14] it was shown that the percentage of time used by HCCA has a significant impact on the resultant QoS. For this reason, we also included the ratio of the time reserved for HCCA as a simulation parameter in addition to traffic types defined above. For more detail about the HCCA and EDCA please refer to [8].

3.2 Simulation with Regression Analysis

The simulation of decision making in 802.11e networks is composed of two periods. The first period is called the training period and the second one is called the decision period. Training period is used for historical data collection and evaluation while the decision period is the verification of the learned cases.

Training period for regression analysis means the parameter estimation procedure which is done as follows. We start one simulation run with a set of traffic streams selected from the traffic types given above. At the 45^{th} simulation second we start measuring the QBSS load element parameters for five seconds. In 802.11e, the duration of time in which QBSS load element measurement is realized is left to vendors. Since 5 seconds proved to be enough for the convergence of the QBSS load element parameters in all cases, we made measurements for the last five seconds. After each simulation run we calculate the MOS values of each voice stream within the run using the measured delay, jitter and loss rates. This gives us the dependent variable of our regression analysis. We also note the station count, channel utilization and available admission capacity during the last five seconds of each run as our independent variables. After running more than 20000 such simulation runs, we make curve fitting using equation (2) and estimate the parameters a,b,c and d.

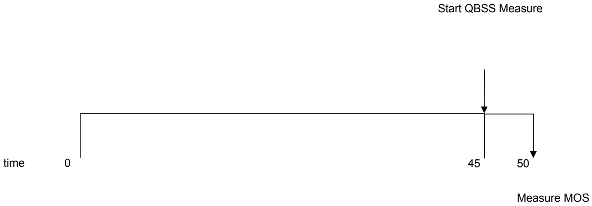


Fig. 1. One simulation run within the training period of the regression analysis

During the decision period we simply use the estimated formula and make decisions for the problems as defined in section 2.3.

3.3 Simulation with Reinforcement Learning

A training step for reinforcement learning for the first problem of section 2.3 is illustrated in figure 2. We can summarize this training step as follows: A number of stations with different traffic types are connected to the access point or access points and start transmitting their packets. Initially the simulation is run for 25 seconds. Afterward, the QBSS load element and the number of traffic streams of different priority levels are found. At this point, the decision making algorithm decides on adding a new voice stream to the simulation or not by considering the information it is given (either the QBSS load element or the number of traffic streams associated with different priority levels). The decision is made either using the previously learned actions with a probability of (1-α) or randomly with a probability of α. The optimum choice of the parameter α is done by making cross-validation analysis together with the parameter γ of the equation (3). For all the results presented in this study we used 0.25 for α and 0.82 for γ, since these values proved to give the best performance. Regardless of the result of the decision, a new simulation with the same configurations is started. At time 25, the new voice stream is joined to the system. After this point, the simulation goes on for 25 seconds and stops at 50 seconds. Jitter, delay, and loss rates are calculated using the last 25 seconds' information and correspondingly MOS values are calculated. If the calculated MOS value is over the threshold which is 3.8 in our case and if the decision of the algorithm was accepting the new traffic, a true decision was given and the algorithm is rewarded directly proportional to the difference between the MOS value and the threshold for MOS. Correspondingly the $Q(s_t, a_t)$ value of the state and the action decided by the algorithm is modified using equation (3). This is one run during training period of the first question of section 2.3. We train the RLA with 1000 such runs.

In fact the number of runs during the training period affects the accuracy of the decisions after training in case the training data does not fluctuate very much [13]. However if there are fluctuations, it makes sense to use less training data to capture the effects of such fluctuations as it will be mentioned in section 5. In our case the learned Q function values converged to a steady state within less than 1000 runs. However we also present results with 100 runs in section 5.

Additionally learning is also dependent on the reward function which is strictly dependent on the MOS value. Cases having MOS values much lower or higher than the threshold value are learned much better than cases with MOS values near to the threshold value. Such configurations of the RLA are problem specific and we won't go into more detail in this study because of size limitations.

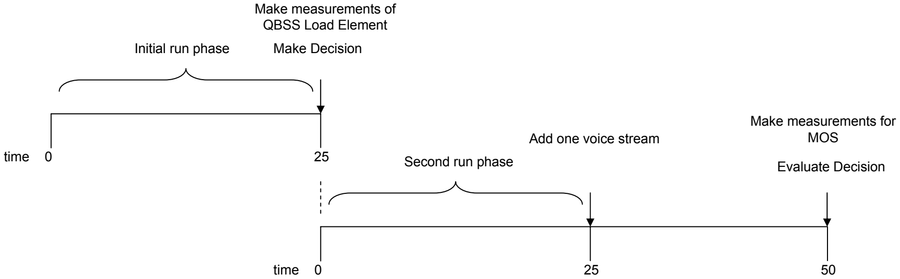


Fig. 2. One simulation run within the training period of the RL algorithm

For the training step of the second question we use two simultaneous simulation runs with two access points. In one of these simultaneous runs, decision is made using the RLA and in the other one decision is always the opposite of the decision given by the RLA. In this way, we can compare the results of opposite decisions and hence decide which action would be accurate. At the end of each run we compare the MOS values of both cases. If an accurate decision is given by the RLA, it is rewarded directly proportional to the difference between the MOS values of both cases. The learning matrix of the RLA is modified by changing the expected values of all related actions with respect to this reward. The $Q(s_t, a_t)$ value of the state and the action decided by the algorithm is modified and a new learning round is started as explained above.

The information learned during the training period, namely $Q(s_t, a_t)$, is used in order to make decisions during the decision period, by simply choosing the action with a greater $Q(s_t, a_t)$ value.

4 Simulation Results

4.1 Results of Regression Analysis

Table 2 summarizes the regression analysis using equation (2) and the information given by the QBSS load element. In our previous research [14] it was shown that the number of traffic streams associated with a priority gives more information than the QBSS load element. Although this information is not available within the standard 802.11e, wireless devices of the same vendor can still make use of this property. For this reason we also applied regression analysis with this information. The corresponding function used during regression analysis including the information about the number of traffic streams of different priorities is:

$$E[QoS] = a + bT_1 + cT_2 + dT_3 + eT_4 \tag{4}$$

where T$_i$ is the number of stations using i^{th} priority. The results of this regression analysis is given in table 3.

Table 2. Linear regression analysis of QBSS load element parameters

	40%HCCA		80%HCCA		98%HCCA	
	Estimate	95% CI	Estimate	95% CI	Estimate	95% CI
a	-4.71	{-5.300,-4.120}	2.588	{2.167,3.010}	-3.205	{-4.11,-2.299}
b	-0.208	{-0.245,-0.171}	-0.195	{-0.220,-0.171}	-0.217	{-0.270,-0.164}
c	13.695	{12.835,14.555}	4.428	{3.798,5.059}	13.237	{11.876,14.598}
d	7.617	{6.717,8.518}	0.672	{-0.178,1.523}	4.146	{2.669,5.623}

Table 3. Linear regression analysis of number of different priorities

	40%HCCA		80%HCCA		98%HCCA	
	Estimate	95% CI	Estimate	95% CI	Estimate	95% CI
a	8.203	{7.893,8.514}	10.759	{10.563,10.955}	8.700	{8.332,9.070}
b	-2.206	{-0.270,-2.143}	-2.927	{-2.959,-2.895}	-2.417	{-2.493,-2.342}
c	-0.242	{-0.322,-0.162}	-0.209	{-0.233,-0.185}	-0.735	{-0.802,-0.669}
d	-0.201	{-0.253,-0.149}	0.000	{-0.038,0.039}	0.045	{-0.019,0.108}
e	0.058	{0.019,0.098}	0.039	{0.012,0.066}	0.003	{-0.042,0.047}

Using the estimated parameters of equations (2) and (4), we run simulations as described in section 3.2. Table 4 and 5 summarize the results.

Table 4. Percentage of true decisions made using regression analysis to find $MOS > thresholdMOS(3.8)$

	QBSS Load Element	number of priorities
40% HCCA	77%	93%
80% HCCA	70%	80%
98% HCCA	55%	79%

Table 5. Percentage of true decisions made using regression analysis to find better access point out of two

	QBSS Load Element	number of priorities
40% HCCA	63%	68%
80% HCCA	60%	65%
98% HCCA	64%	64%

Our first observation within tables 4 and 5 is that the percentage of true decisions decreases if the percentage of HCCA increases. This supports our claim in [14] that because of the complexity of HCCA with respect to EDCA, estimating the quality of service by only considering the QBSS load elements is very difficult. Additionally we see in both tables that having the number of traffic streams of each priority improves the percentage of true decisions in all cases. On the other hand, if we compare the results of table 4 with the results of table 5, the percentage of true decisions decreases if the problem is choosing the better access point out of two instead of accepting an access point with an expected MOS value bigger than the given threshold.

Although the set of the data used during the decision making phase and the training phase of the simulations were the same for regression analysis, which brings significantly less dynamism to the problem, the best results we had using QBSS load element was 77% as seen in tables 4 and 5. On the other hand this percentage drops to 55% if nearly all the time is reserved for HCCA. Nevertheless, if the number of traffic streams of different priorities is known, the decision correctness is significantly improved and can be as high as 93%, which is an acceptable level. However the decision accuracy is very low when two access points are compared. This is mostly due to the fact that, although regression analysis can estimate the MOS for a specific state to some extent, this estimation is not exact enough to distinguish two similar states.

Considering the given results, we come to the conclusion that using simple logic such as, more channel utilization means less expected QoS for new traffic streams or correspondingly a simple way of decision making like curve fitting cannot help making reliable decisions for choosing an access point which also means that there is no straightforward relationship between the QBSS load element parameters and the resultant QoS level.

4.2 Results of Reinforcement Learning

The results of reinforcement learning are given in tables 6 and 7. As seen from tables 6 and 7, knowing the number of traffic streams of different priority levels allows more accurate decisions during association with an access point. Additionally although the difference is relatively lower, more time reserved for HCCA decreases the percentage of true decisions.

Table 6. Percentage of true decisions made using reinforcement learning to find $MOS > thresholdMOS(3.2)$

	QBSS Load Element	number of priorities
40% HCCA	88%	97%
80% HCCA	81%	94%
98% HCCA	79%	95%

Table 7. Percentage of true decisions made using reinforcement learning to find better access point out of two

	QBSS Load Element	number of priorities
40% HCCA	84%	92%
80% HCCA	71%	94%
98% HCCA	67%	96%

If we compare the performances of regression analysis and the RLA, it is clear that reinforcement learning performs in all cases better than regression analysis. In fact such kind of a result is not strange. Regression analysis tries to optimize its estimation considering all the states of the input parameters. For this reason, especially neighboring states determine the slope of the resultant curves. Reinforcement learning deals mainly with individual states which optimizes the decision for a state apart from all other states. If the behavior of the whole system cannot be described with a convex or concave system, it is more probable that such algorithms like RLA perform better than regression analysis. During our analysis we tried many combinations of linear and non-linear equations. However, none proved to perform better than the linear regression. Although this does not mean that such an equation does not exist, alone this difficulty in finding a suitable equation representing an environment makes the use of RLA more attractive.

5 Deploying Reinforcement Learning over Wireless Devices

In the previous section we showed that the performance of RLA is significantly higher than regression analysis which includes but not restricted to the traditional ways of candidate access point selection algorithms. Hence it makes sense to use reinforcement learning over wireless devices of 802.11e. However reinforcement Learning is a relatively complex algorithm compared to the static way of decision making over wireless devices. For this reason it needs significantly higher hardware resources during runtime. Nevertheless we can overcome this obstacle by doing simple adjustments in the way of using reinforcement learning.

Reinforcement learning is composed of two periods, the learning period and the period in which the learned actions are used. In fact in terms of CPU time, the only challenging part is the training period since we need a high number of training steps which is in our case 1000. However it is not logical to expect that a mobile station has to wait 1000 connections in order to complete a learning procedure. Instead of that, vendors of wireless devices can give a trained Q function (See section 2.2) to the RLA. Then the problem is reduced to two subproblems. First is the decision making with an already trained Q function and the second is the adaption of the reinforcement algorithm to a new environment. The first problem is no more different than decision making in a static way, since access points just have to check the value of a state in the Q function for making a decision. Consequently we do not have a CPU time problem. The latter is the most interesting point, since it distinguishes reinforcement learning from the static way of decision making. For this purpose we tested RLA by using a trained Q function and exposing the mobile stations to a highly dynamic environment in order to see if the Q function could be adapted with respect to new conditions successfully. Figure 3 shows the results of this analysis.

In figure 3, a positive Q function value means a positive answer to the access point and a negative value a negative answer. If the Q value is decreasing, then

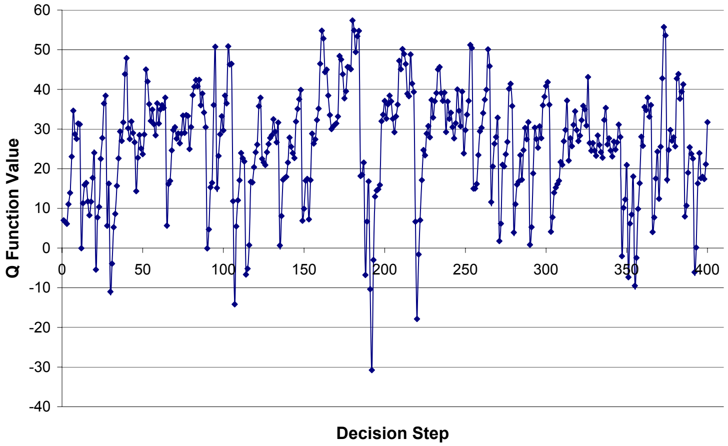


Fig. 3. An example to RLA adaptation within a dynamic environment. Points represent the changing Q function value of state $s_t(2,3,74)$ at each decision step after rewarding RLA.

either the reward received after completing one step was not enough to compensate the decrement in the Q function value because of the discount factor γ, or there was a negative reward meaning that the decision was false. As it can be seen from figure 3, in nearly all of the cases where the sign of the Q function value changes from positive to negative we need less than 5 steps to change the sign. This means the correct choice is learned within less then 5 steps. In fact we consider 5 steps to be negligible for a capability of dynamic decision making. Consequently the use of reinforcement learning proves to be significantly efficient for dynamic decision making for the candidate access point selection problem.

6 Conclusion

The desire for a more autonomic communication environment which would work with any technology, protocol or service and without the need for intervention from the humans is growing. Such an environment should ease the way communication is being done and also maximize QoS by considering its own situation dynamically. However, the research in this field is still in its initial phase, where even a reliable and widely accepted modeling of autonomic communication is missing. On the other hand, introduction of new technologies such as 802.11e ease reaching this target, since they allow the deployment of more intelligent decision making algorithms and enable QoS negotiation among peers dynamically.

In this paper we analyzed the use of a dynamic decision making algorithm based on reinforcement learning for solving the candidate access point selection problem of 802.11e networks. We compared its results with the traditional way of decision making using regression analysis. We showed that if the stations are lacking information about network parameters which do have high correlation rates with the quality of service metrics, then only more complex algorithms like RLA can be used as a reliable solution for the candidate access point selection problem.

During our simulations, it was possible to have up to 77% correct decisions using regression analysis in case we want to know if the QoS will be over a given threshold. However this percentage fell down to 55% when we compared two access points with a relatively higher percentage of time used for HCCA. Such low levels of correct decisions showed that it is not straightforward to confirm a one to one relationship between the ”QBSS load element” parameters and the quality of service metric - here MOS. On the other hand, the use of RLA which optimizes its decisions by dealing with individual states apart from the other states proved to perform much better. We reached up to 89% correct decisions if the question was finding QoS levels higher than a threshold. This result dropped to 67% when comparing QoS of two access points with higher percentage of time reserved for HCCA. As a possible enhancement to the QBSS load element, we showed that the use of the number of traffic streams of different priorities improved the performances of both algorithms considerably. Even regression analysis reached up to 90% correct decisions having only minimal differences with the performance of the reinforcement learning algorithm.

Although RLA introduces the need for more hardware resources, we showed that this is not the case if an already trained algorithm is used initially. We demonstrated that, in this way the use of the reinforcement learning algorithm during decision making can be reduced to a static way of decision making which overcomes the problem of real time application of the algorithm that might occur because of scarce hardware resources. Additionally we illustrated that RLA can adapt itself to new environments by trial and error within less than 5 false decisions within a highly dynamic environment. This introduces a substantially higher advantage when compared with the traditional way of decision making. Our next study is going to deal with the enhancement of the reinforcement learning algorithm for admission control purposes using the traffic specification element and the ADDTS service primitive of IEEE 802.11e.

References

1. ITU-T Rec. P.85. A Method for Subjective Performance Assessment of the Quality of Speech Voice Output Devices, 1994.
2. ITU-T Rec. G.107. The E-Model, a Computational Model for Use in Transmission Planning, 2002.
3. P. Ansel, Q. Ni, and T. Turletti. An Efficient Scheduling Scheme for IEEE 802.11e. In *Proceedings of IEEE Workshop on Modelling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, March 2004.
4. P. Ansel, Q. Ni, and T. Turletti. FHCF: An Efficient Scheduling Scheme for IEEE 802.11e. In *ACM/Kluwer Journal on Mobile Networks and Applications (MONET), Special Issue on Modelling and Optimization in Wireless and Mobile Networks*, August 2005.

5. M. Asada, E. Uchibe, S. Noda, S. Tawaratsumida, and K. Hosoda. A Vision Based Reinforcement Learning for Coordination of Soccer Playing Behaviours. In *Proceedings of AAAI-94 Workshop on AI and A-life and Entertainment*, 1994.
6. J. Baxter, A. Tridgell, and L. Weaver. KnightCap: A Chess Program that Learns by Combining TD(lambda) with Game-tree Search. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.
7. Fiberlink. General Market Statistics. http://www.fiberlink.com /release/en-US/Home/KnowledgeBase/Resources/Stats/.
8. IEEE. *802.11E-2005 IEEE Standard for Information technologyTelecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements*, November 2005.
9. J. Koza. The genetic programming paradigm: Genetically breeding populations of computer programs to solve problems. In B. Soucek and the IRIS Group, editors, *Dynamic, Genetic, and Chaotic Programming*, pages 203–321. John Wiley, New York, 1992.
10. F. Kozamernik. Media Streaming Over the Internet an Overview of Delivery Technologies. Technical report, EBU, October 2002.
11. D. MacKay. Cambridge University Press, 2003.
12. C. Na. *IEEE 802.11 Wireless LAN Traffic Analysis: A Cross-layer Approach*. PhD thesis, The University of Texas at Austin, May 2005.
13. B. Simsek, S. Albayrak, and A. Korth. Reinforcement Learning for Procurement Agents of the Factory of the Future. *IEEE Congress on Evolutionary Computation Proceedings*, 2004.
14. B. Simsek, K. Wolter, and H. Coskun. Analysis of the QBSS Load Element Parameters of 802.11e for a priori Estimation of Service Quality. *International Journal of Simulation: Systems, Science and Technology, Special Issue: Performance Engineering of Computer and Communication Systems*, 2006.
15. R. Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, University North Carolina Charlotte, 2003.

A Multi Agent System Approach for Self Resource Regulation in IP Networks

Gérard Nguengang[1, 2], Louis Hugues[1], and Dominique Gaiti[3]

[1] R&D Ginkgo-Networks, 8 Rue du Capitaine Scott 75015 Paris, France
[2] Laboratoire d'Informatique de Paris 6, 8 Rue du Capitaine Scott 75015 Paris, France
[3] Charles Delaunay Institute – environment of autonomous networks team,
University of Technology of Troyes, 12 rue Marie Curie, BP 2060, 10000 Troyes, France
{gnguengang, lhugues}@ginkgo-networks.com,
dominique.gaiti@utt.fr

Abstract. As Internet is becoming the global infrastructure for all media communications, ensuring the reliability and the quality of network services is far from being a secondary task. Indeed, many business activities rely on the network and any performance degradation can result in serious financial losses. The main objective of Internet Service Providers is to maintain permanently an acceptable service delivery for their subscribers and respect scrupulously the Service Level Agreements. To achieve this goal, just committing bandwidth resources is not sufficient since network services are not safe from breakdowns. The dysfunction of some of its elements or persistent overloads generated by bursts traffics can induce serious deterioration of the network performance and impact the end users' applications. Continuous service monitoring is then required and an autonomous network resource regulation mechanism is necessary in such situation to avoid the degradation and the collapse of all the applications sharing the impacted resource. In this paper, we propose a decentralized approach for the monitoring and management of the network backbone shared resource by the mean of distributed autonomous agents. The agents are deployed at the edge routers and cooperate together in order to maintain a global acceptable level of service in critical situations where the current quality of service is less than the expected one. This allows self adaptive service and resource management with an interesting abstraction from network backbone heterogeneity and complexity.

Keywords: Agents, Resource management, Network performance, Active probing.

1 Introduction

Due to the emergence of multimedia applications such as real time voice and video over Internet and the increase of business dependence on IT infrastructures, computers networks are involving to support services with diverse performance requirements. This is possible thanks to new scheduling algorithms that allow differentiated packets treatment and offer thus several forwarding alternatives. Initially conceived to provide the single best effort service, Internet is actually a multi

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 64 – 75, 2006.

service network. Internet Service Providers (ISP) are now offering a portfolio of services with guarantees in terms of end-to-end delay, jitter, and packets loss. Subscribers can choose among different levels of Quality of Service in order to best meet their applications and pricing constraints. The service and its delivery quality are negotiated through a contract, the Service Level Agreement (SLA).

However, to ensure reliability and good quality of service on these networks is not an easy task. The commitment of network resources is not sufficient to eliminate possible performance degradation. Indeed, persistent overloads in the current Internet is unavoidable [1] and can arise for several reasons: a single flow not using end-to-end congestion control and continuing to transmit despite encountering a high packets drop rate (UDP flows), the dysfunction or outage of a network element, etc… When such a situation occurs, all flows crossing the overloaded resource experienced significantly degraded service over an extended period of time. This is unacceptable since any SLA violation can lead to severe performance degradation and generate serious financial losses for those subscribers whose business activities rely on the network. Therefore, continuous performance monitoring is required to tract the ongoing QoS, compared the monitored QoS with the expected performance, detect possible QoS degradation and then take correctives actions accordingly to sustain as much as possible the agreed QoS for at least critical data. With the vastness of the current networks, the permanent monitoring of each network element in order to track any performance degradation proves to be expensive and sometimes inadequate. Our approach for solving this problem is to consider the network backbone as a shared resource among all the end users and integrate in each edge router a software agent responsible of the monitoring of the end-to-end quality of service and the execution of correctives actions in case any anomaly is detected. The agents cooperate together to coordinate their actions. Our Multi agent architecture makes it possible to ensure proactively the respect of some SLAs when, for an unforeseeable reason, a network service is unable to satisfy all the QoS requirements of its traffics.

This paper proposes a Multi Agent System approach for self adaptive resource management. The aim is to maintain an acceptable level of service delivery when a performance problem occurs in the network with an interesting abstraction from the network backbone heterogeneity and complexity. The organization of the paper is as follows. Section 2 provides a survey of related research in the area of adaptive network resource management. Section 3 gives an overview of our approach. Section 4 describes in details the agent architecture. The experimental setup used to evaluate the concept and some results are presented in section 5. Section 6 summarizes our proposal and addresses future works.

2 Related Works

Performance management is nowadays a key issue in the network management process. Network management systems must evolve to allow fast anomaly detection and rapid problems recovery in order to maintain a good level of service and therefore the respect of the SLA for all the applications using the network. Several researches were undertaken during these last years to propose solutions for an adaptive performance and resource management in IP Networks. The aim of this section is to present existing works in the field of adaptive resource management.

The current tendency in network management is the self-aware management, that is, enabling the management processes and the subjacent infrastructure to organize themselves and operate without external assistance [2]. It includes self-configuration, self-optimization, self-healing and self-protection. In terms of resource management, increasingly sophisticated techniques are needed to improve the overall service quality and minimize the effects of potentially damaging periods of poor service [3]. Vilà and al [4] have proposed a dynamic bandwidth management scheme in logical network such as ATM or MPLS based on distributed agents. The system consists in two types of agents: The Network Monitoring agents (M) and the Network Performance agents (P). Each node has one P agents and several M agents. The M agent monitors and controls a single logical path (LP). P agents are responsible of the supervision and the collaboration of the M agents of the same node and cooperate with its peers to perform bandwidth management and re-routing of the LPs. For instance, if a LP is congested, its M agent will detect the congestion (threshold violation) and warn the P agent. The affected P agent may send a message to one of its neighbors requesting some action. The P agent that receives the message firstly merges the partial network view and then, it uses its actualized partial network view to make a decision or take an action. [5] also proposed a decentralized architecture of autonomous, collaborative agents that can model normal network operations, detect departures from expected behaviors, and take remedial actions to avoid performance degradations. Each agent creates a thumbprint of network behavior under normal circumstances. Agents compare these thumbprints to observe performance and detect deviations from normal activities. Although the aforementioned proposals enhance the control and the management of network resources and allow fast recovery of network performance when network anomalies occur, their implementation requires the “agentification” of all the network elements since all the nodes are involved in the management process. Also, the network monitoring is not service-oriented. Even if no performance degradation is detected, these systems cannot ensure that the network services QoS requirements are respected.

[6, 7, 8] have proposed measurement based admission control (AC) algorithms for efficient network resource management. A new flow is admitted in the network if the current network load permits the provision of its QoS requirements. In [9] a distributed service-oriented traffic control mechanism based on online active monitoring is proposed. AC decisions are made based on feedback from edge-to-edge on-line measurements of service specific QoS parameters. These proposals present a very attractive solution because there is a real possibility of making reasonable estimates of the properties of end-to-end paths from edge-based measurements. However, if a network service experiences suddenly performance degradation, all the flows using it will be impacted.

3 Overview of the Approach

In an ideal world, the network should be able to proactively detect service disruption or performance degradation and freely take corrective actions to overcome or minimize their effects on network end users applications without any human intervention and in the respect of the SLAs. This is impossible with the classic centralized network management paradigm based on the periodical polling of network devices. The amount of information and computation needed to estimate in real time the network state is huge and the process of data aggregation, treatment and decision can take too much time. It is thus important to divide the problem for better solving it. Multi agent system paradigm provides a decentralized approach to solve difficult problems in complex environments. One of the main ideas of multi agent system is to generate approximate solutions to hard problems by distributing them to autonomous rational problem solvers (agents) that have local problem solving capabilities and are able to find a solution for the whole problem by cooperating with each other [10].

In our approach, we combine the use of distributed agents and the active service monitoring to regulate the network utilization. We make the assumption that the network backbone is a shared resource among all the end users. This has the advantage of providing an abstraction of the network core complexity. Figure 1 gives an overview of the system.

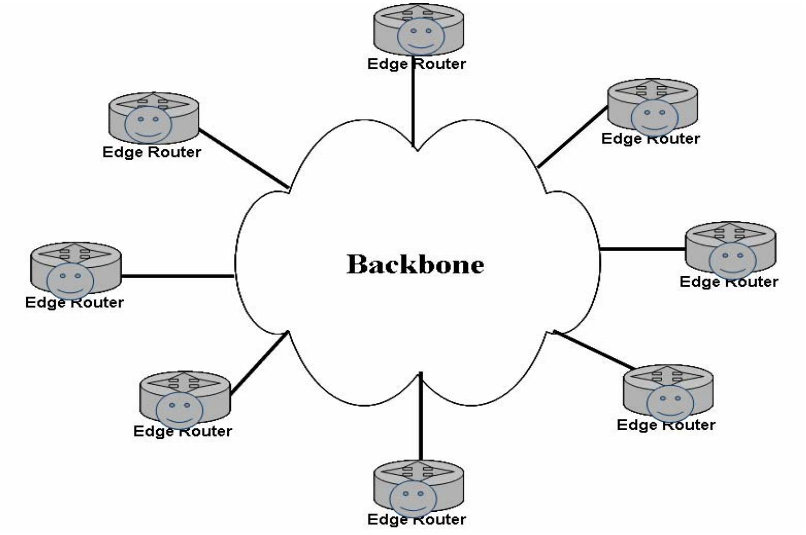


Fig. 1. An Overview of the approach

Each edge router is equipped with an agent. The agent carries out the on-line monitoring of the network services performance. Periodically, on the basis of its measurements, each agent computes a local indicator. This indicator represents the agent perception of the network backbone health, a situated representation of the network core state. This information is thereafter broadcasted to the other agents of the multi agent system. Once the indicators are collected, the agents individually start the diagnosis process and evaluate if an action must be taken or not. If the current network performance requires an adaptation, the agents cooperate together to define the action to undertake and elect the agent responsible of its execution. This is to avoid unilateral agents' answers to a problem which could lead the network to a state of under utilization.

4 The Agent Architecture

Jennings & al [11] define an agent as an entity which is:

- *Situated* in some environment.
- *Autonomous*, in the sense that the system can act without direct intervention from others (humans or other software processes).
- *Flexible*, which is further broken down into three properties: *responsive* (perceives its environment and responds to changes in a timely fashion), *pro-active* (exhibits opportunistic, goal-directed behaviour) and *social* (able to interact with humans or other artificial agents).

This definition corresponds to the capacities we intend to equip our agents. They must be able to act on the control plane of their router in a coordinated way. To achieve this goal, each agent is made up of three fundamentals elements: the knowledge base, the situated network view, and the behaviors. Figure 2 summarizes our agent architecture.

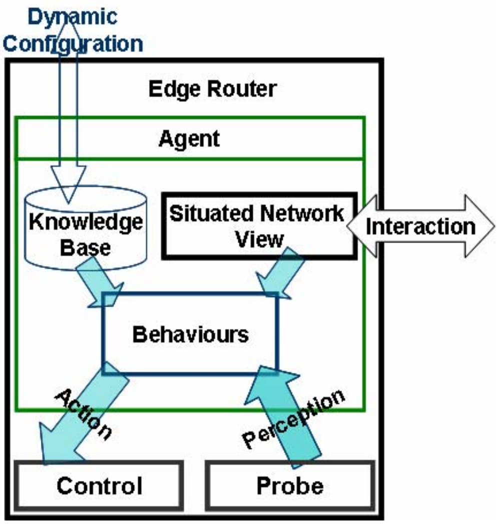
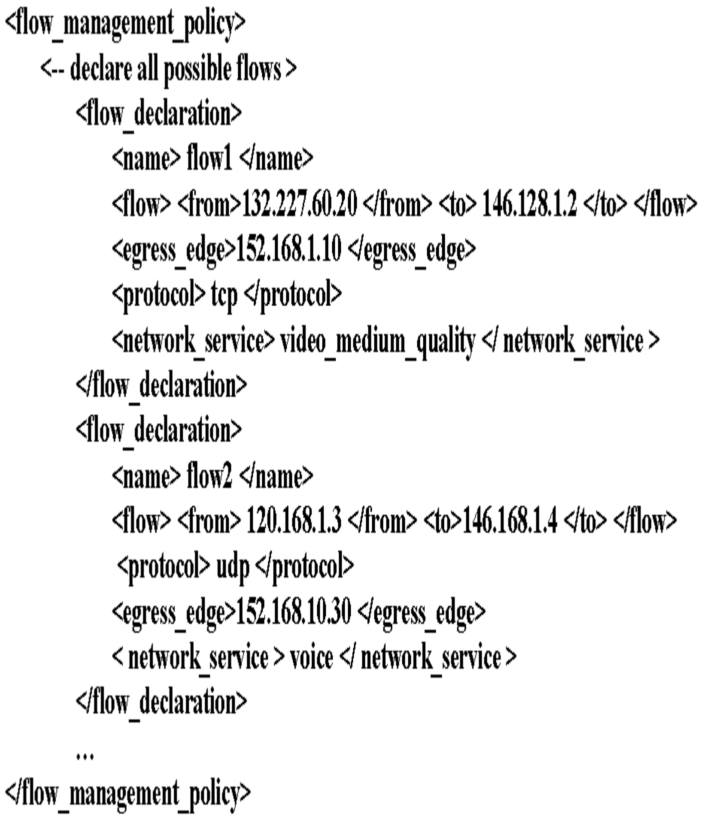


Fig. 2. Agent Architecture

4.1 The Agent Knowledge Base

The goal of the agent is to monitor the services provided by its edge router and cooperate with its peers to take corrective actions when a performance anomaly occurs. The agent knowledge base contains a description of all the flows entering the network by its edge router and a specification of the services offer by the network. A flow is characterized by the IP source address, the transport protocol, the destination port, the egress edge IP address and the network service used for its transport. Figure 3 shows the XML description of a flow.



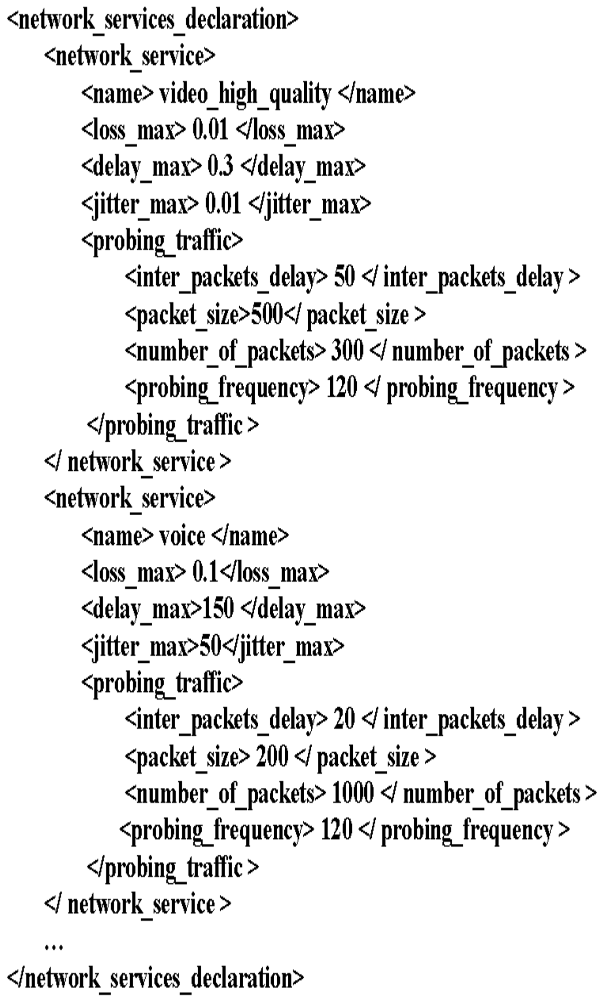
```
<flow_management_policy>
    <-- declare all possible flows >
        <flow_declaration>
            <name> flow1 </name>
            <flow> <from>132.227.60.20 </from> <to> 146.128.1.2 </to> </flow>
            <egress_edge>152.168.1.10 </egress_edge>
            <protocol> tcp </protocol>
            <network_service> video_medium_quality </ network_service >
        </flow_declaration>
        <flow_declaration>
            <name> flow2 </name>
            <flow> <from> 120.168.1.3 </from> <to>146.168.1.4 </to> </flow>
             <protocol> udp </protocol>
            <egress_edge>152.168.10.30 </egress_edge>
            < network_service > voice </ network_service >
        </flow_declaration>
        ...
</flow_management_policy>
```

Fig. 3. Example of XML description of a flow

The network service specification includes QoS requirements in terms of authorized maximum value of the delay, jitter and packets loss as well as the characterization of the traffic which will be used for probing. The idea is to send test traffic corresponding as much as possible to the behavior of the ingress traffics in order to have the most exact estimation of the network response time. On the basis of its probing results, the agent will evaluate if the network is able to deliver the awaited services. Figure 4 shows the XML description of network services.

4.2 The Situated Network View

The IP Network backbone is not a static entity. The network state is frequently prone to fluctuations due to the high variability of the traffics generated by the users' applications. It is thus very difficult to define a model of the network behavior. That is why the agent has to build its own representation of its environment. This operation is done thanks to the agent communication module. The agent receives from its peers their perception of the current network backbone quality of service. All those informations constitute the situated network view of the agent. They are crucial for the agent decision-making process.



```
<network_services_declaration>
    <network_service>
        <name> video_high_quality </name>
        <loss_max> 0.01 </loss_max>
        <delay_max> 0.3 </delay_max>
        <jitter_max> 0.01 </jitter_max>
        <probing_traffic>
            <inter_packets_delay> 50 </ inter_packets_delay >
            <packet_size>500</ packet_size >
            <number_of_packets> 300 </ number_of_packets >
            <probing_frequency> 120 </ probing_frequency >
         </probing_traffic >
    </ network_service >
    <network_service>
        <name> voice </name>
        <loss_max> 0.1</loss_max>
        <delay_max>150 </delay_max>
        <jitter_max>50</jitter_max>
        <probing_traffic>
            <inter_packets_delay> 20 </ inter_packets_delay >
            <packet_size> 200 </ packet_size >
            <number_of_packets> 1000 </ number_of_packets >
           <probing_frequency> 120 </ probing_frequency >
         </probing_traffic >
    </ network_service >
    ...
</network_services_declaration>
```

Fig. 4. Example of XML description of networks services

The knowledge base can be updated dynamically by a remote system.

4.3 The Agent Behaviors

Successively, the agent carries out two behaviors:

- the Service Probing behavior
- the Flow Control behavior

4.3.1 The Service Probing Behavior

The Service Probing behavior allows the monitoring of the QoS requirements for all the defined network services. The QoS monitoring involves the definition of metrics, measurement methodology and timing decision. The IETF IPPM defined a set of standard QoS and performance metrics and proposed measuring methodologies for them [12, 13]. For each service, the probe generates test traffics in direction of all the edge routers and computes the QoS metrics. This traffic is generated in conformity with the specifications contained in the network services declaration XML file. In this work, with an aim of simplification, we decide to restrict the QoS metrics to the end-to-end delay between edges routers. This end-to-end delay is computed by dividing the round trip delay by two. It is impossible without any additional information to know which direction generates the delay. It is certainly a coarse estimation of the one-way delay but it freed from the complexity of the tools necessary for an exact measurement. Hence, the measurement of the one-way delay requires the deployment of Network Time Protocol (NTP) or Global Positioning System (GPS) for the end-points clock synchronization. Once that the edge-to-edges delays are measured for each service, a local indicator for each service is then computed. The indicator varies between 0 and 1. The computing process of the local indicator i_S for service S is as follows:

```
Step one:

Extract the highest delay for the service S and the
agents involved in its measurement e.g measured delay
D, from Agent A to Agent B = (D,A,B) where D is the
highest delay measured for the service S

Step two: Indicator computation

if (measured_Delay<= Max_Athorized_Delay)

         $i_s$ = (measured_Delay / Max_Athorized_Delay)

else     $i_s$ = 1
```

When, for a service, the indicator ***$i_S = 0$,*** nothing is wrong on it. Oppositely, when ***$i_S = 1$,*** it implies that the flows using the service are in trouble. Something must be done to reduce to rectify the situation.

The computed indicators are broadcasted by the agents to their peers in the multi agent system.

4.3.2 The Flow Control Behavior

The flow control behavior consists in the decision-making process and the execution of the adequate actions in order to adapt the network traffic according to the available network resources and avoid the complete collapse of network services. The Flow Control behavior takes in input the set of indicators available in the network situated view and checks for each service if it is necessary to initiate a coordinated action. The possible actions are either authorizing traffic to enter the network backbone or prohibiting the access of a flow to it. This is realized thanks to the agent control module which acts directly on the routing functions of the router OS. For each service, the first step of the decision making process is to determine the edge router that appears more than the others in the destination field of the indicators. This indicates that the concerned destination is in serious trouble. If the greatest of these indicators exceeds a certain threshold, a message is sent to all the agents which some ingress flows are in direction to the previous detected destination. A decision must be taken to release the resource and avoid persistent congestion and performance degradation. It will be taken by one of the agents having received the message according to whether it is the smallest in the collating sequence, each agent being identified by an alphabetic letter.

5 Experimental Setup

A test-bed has been carried out to test our concepts and evaluate the relevance of our approach. Figure 5 shows the detailed setup. It consists of:

- 4 PC-based Linux routers
- 2 video servers. The VideoLAN software is used for the video streaming and viewing.
- 2 video clients

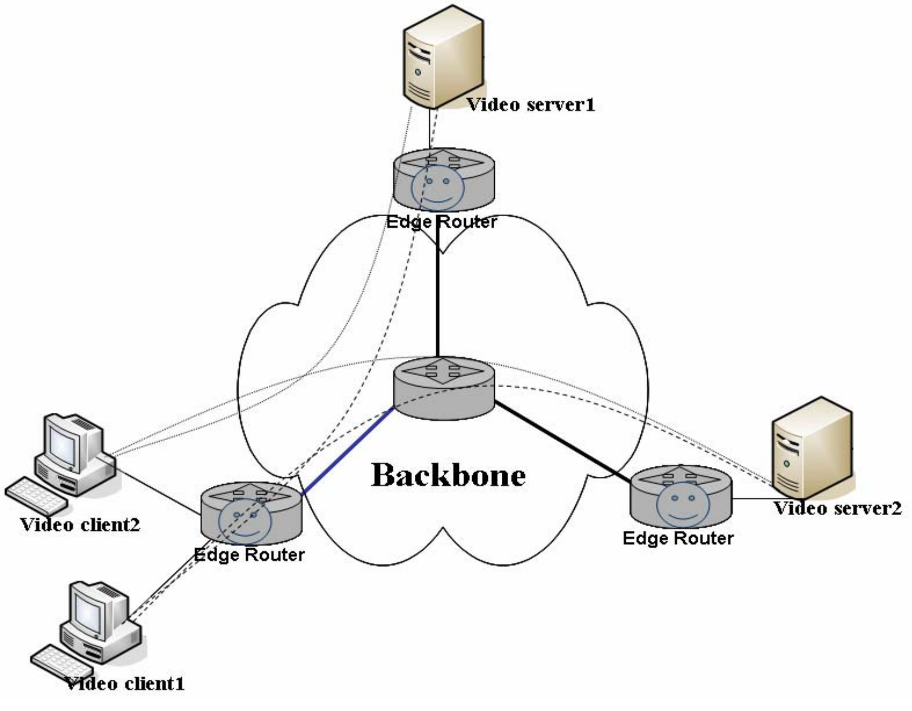


Fig. 5. Test bed

To simplify the tests, we considered that the network provides only one service, the video service. The authorized maximum delay for the video service is fixed at 500 ms. The agents are implemented in Java and the probe module in C. The test scenario is as follows:

2 video flows are streamed in loop from the servers to the clients on the udp protocol (figure 5). The incoming flows properties are declared in the knowledge base of the corresponding agent. Thus on each screen, it is possible to see two different videos. In a first stage, sufficient network resource is provided. All the links are point-to-point 100Mbit Ethernet. The videos have a perfect viewing quality. No degradation is observed on the images. Then, to simulate a performance problem on the network, the bandwidth of the link which connects the video clients' edge router to the backbone is reduced to 7.600Mbit. This is done thanks to the token bucket filter queue discipline of the Linux traffic controller module. With this bandwidth, the streams start to experience degradations. The video service is impacted by the lack of bandwidth. The images become fuzzy on the screens. The multi agent system is not yet activated. So, if nothing is done to reestablish the initial status, all the end-users' video applications will suffer of this performance degradation. Under these conditions, the agents are activated within the edge routers. The lack of sufficient network resources is detected by the system and an agent is elected to prohibit randomly the access to the network of one of its incoming flows. This has as a result the improvement of video service delivery and a good visual quality for the remaining streams. Figure 6 shows the delay variations of the video service between the video servers' edges routers and the customers' edge router some time before the agents' activation and after. We notice that before the agents' activation, the transfer delay of the video service is very high and oscillates between 700 ms and 1000 ms. Once the multi-agent system is activated, this delay falls and approaches zero. This is due to the

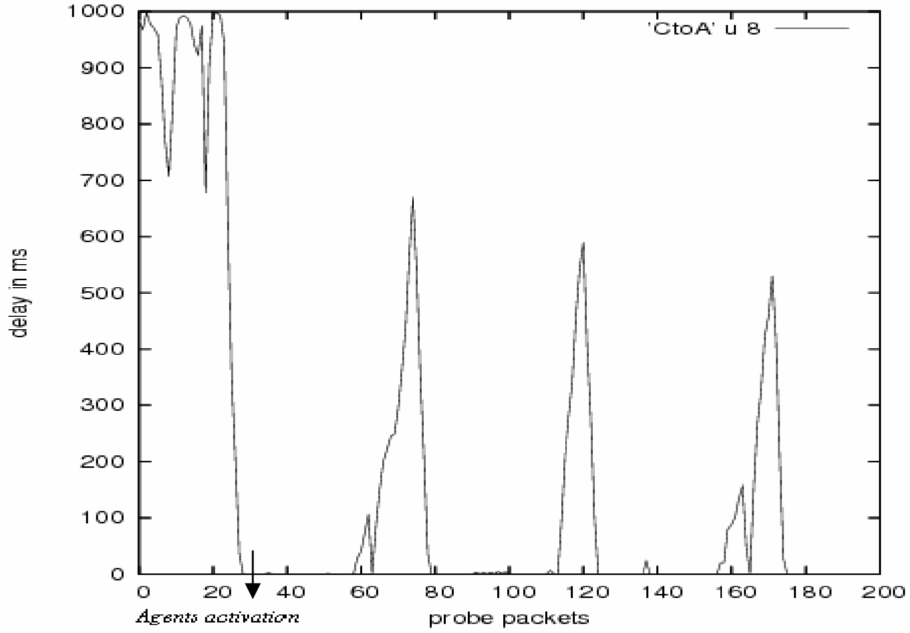


Fig. 6. Network delay variation

fact that one of the ingress streams is stopped and the shared resource is not any more congested. At the end of a variable time, since it depends on the network state, the system estimates that the anomaly having caused the performance degradation and the interruption of one of the flows is solved. This estimation is effective when the time series average of the ten last service indicator measurements is under the low threshold. The agents cooperate together and authorize suspended flows to reach again the network. Since the bandwidth limitation still exists, the system reacts and stops a video stream. This explains the jumps of delay observed thereafter.

6 Conclusion and Future Works

The unforeseeable breakdowns or anomalies when they occur in the network very often cause a considerable degradation of its quality of service. In these cases, all flows using the impacted services undergo the effects of the lack of sufficient network resource. That has as a consequence the possible violation of all the SLA of the customers using these services and possible financial losses for the ISP. To mitigate this perverse effect, it is necessary to implement a self-regulation system to control and regulate in near real time the amount of traffic to be admitted in the network according to the available resources. In this paper, we have specified a multi agent system in which every agent probes the network backbone and cooperates with its peers to finally authorize or prohibit the access to some ingress traffics. This allows more dynamicity in the control of network services and adaptive resource management while abstracting from network complexity and heterogeneity.

The practical experiments have permitted to prove the viability of our approach. Future works intend to extend our investigation to more complex scenario with several network services (voice, video, file transfer), estimate more precisely the impact of the test traffic on the network performance and evaluate the scalability of our solution.

References

1. R. Mahajan, S. Bellovin, S. Floyd, J. Loannidis, V. Paxson, S.Shenker, Controlling High Aggregate in the Networks, Technical report , February 2001
2. F. Krief, Self –aware management of IP networks with QoS guarantees, International Journal of Network Management 2004; 14: 351-364.
3. S. Willmott, M. Calisti, An Agent Future for Network Control? Revue des organisations suisses d'informatique, INFORMATIQUE 1/2000
4. P.Vilà J.L Marzo, E. Calle, Dynamic Bandwidth Management as part of an Integrated Network Management System based on Distributed Agents, Globecom 2002
5. L. J.Gash, Scalable autonomous monitoring and response for network communication reliabity, in proceedings of The 2004 Military Communications Conference, MILCOM '04, Monterey CA.
6. S. Jamin, P. B. Danzig, S. J. Shenker, and L. Zhang, "A measurement-based admission control algorithm for integrated services packet network", IEEE Trans. Networking., 5(1), Feb. 1997, pp. 56-70.

7. L. Breslau and S. Jamin, “Comments on the performance of measurement-based admission control algorithms”, IEEE Infocom 2000, Tel Aviv, Israel.
8. J. Qiu and E. W. Knightly, “Measurement-based admission control with aggregate traffic envelopes”, IEEE/ACM Trans. Networking, vol. 9, no. 2, April 2001.
9. S. Rito Lima, P. Carvalho, V. Freitas, Self-adaptive Distributed Management of QoS and SLSs in Multiservice Networks, 9th IEEE/IFIP International Symposium on Integrated Network Management (IM'2005) (Session 9) IEEE Press, Nice, France, May 15-19, 2005
10. K. Ficher, C. RuB, G. Vierke, Decision Theory and Coordination in Multiagent Systems, Reseach Report, September 98.
11. N. R. Jennings and M. A. Gibney, Dynamic Resource Allocation by MArket-Base Routing in Telecommunications Networks. In S. Albayrak and F. J. Garijo, editors, Proceedings Second International Workshop on Intelligent Agents for Telecommunications Applications IATA’98, pages 102-117. Springer (as LNAI-1437), 1998
12. G. Almes, S. Kalindindi, and M. Zekauskas, A One Way Delay Metric for IPPM, IETF RFC2679, 1999
13. IPPM-WG, IP Performance Measurements Working Group, http:/www.ietf.org/html.charters/ippm-charter.html.

DoS Protection for a Pragmatic Multiservice Network Based on Programmable Networks*

Bernardo Alarcos[1], María Calderón[2], Marifeli Sedano[3], and Juan R. Velasco[1]

[1] Department of Automática, Universidad de Alcalá, Madrid, Spain
{bernardo, juanra}@aut.uah.es
[2] Department of Ingeniería Telemática, Universidad Carlos III de Madrid, Madrid
maria@it.u3cm.es
[3] Department of Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Madrid, Spain
marifeli@gsi.dit.upm.es

Abstract. We propose a scenario of a multiservice network, based on pragmatic ideas of programmable networks. Active routers are capable of processing both active and legacy packets. This scenario is vulnerable to a Denial of Service attack, which consists in inserting false legacy packets into active routers. We propose a mechanism for detecting the injection of fake legacy packets into active routers. This mechanism consists in exchanging accounting information on the traffic between neighboring active routers. The exchange of accounting information must be carried out in a secure way using secure active packets. The proposed mechanism is sensitive to the loss of packets. To deal with this problem some improvements in the mechanism has been proposed. An important issue is the procedure for discharging packets when an attack has been detected. We propose an easy and efficient mechanism that would be improved in future work.

Keywords: Active Networks, Security, Denial of Service.

1 Introduction

Active and programmable networks [1] facilitate the provision of new dynamic services, introducing programmability into some nodes. We propose the use of an active router based on the SARA[1] platform [2] to build a pragmatic multiservice network.

The users of the multiservice network can request services (e.g. caching, transcoding of multimedia flow…) to improve the communications between the end system users and other end systems in the network. To offer a service, active routers execute some specific codes to process packets exchanged between end systems. The multiservice network requires security services to guarantee that only authorized users can deploy services.

* This work has been funded by CICYT under project IMPROVISA (TSI2005-07384-C03).

[1] SARA (Simple **Active Router**-Assistant Architecture) is a prototype of an active router developed under the GCAP IST project by the active networks researchers group of the Carlos III University of Madrid.

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 76 – 85, 2006.

Active routers are the critical point that must be protected. The active routers execute dynamic codes to process active packets carrying control information and legacy packets carrying data. The active packets must be authenticated to avoid fake active packets changing the behaviour of active routers. We have proposed a security solution to tackle these problems in [3]. Other problem can appear when fake legacy packets, which are also processed by active routers, are injected into active routers. A Denial of Service (DoS) attack occurs in active routers in this situation. In this paper we describe a solution to tackle this problem, which consists in exchanging accounting information between the neighbouring active routers. The insertion attacks can be detected at the attacked nodes by comparing the exchanged information. A reaction mechanism to use when an attack occurs is also defined.

The rest of the paper is as follows: in section 2 we describe a proposal for the multiservice network, in section 3 the security problem in this scenario is presented, in section 4 we propose a mechanism to solve the problem, in section 5 some validation tests are presented, and finally section 6 is devoted to the conclusion and future work.

2 A Pragmatic Vision of Multiservice Networks Based on Programmable Networks

The multiservice network that we propose is made up of a number of active routers within an IP network. The active routers identify special packets called active packets and load a specific code to process these active packets. Active packets go from an end system (source) to an end system (destination), and the active routers in the path between the source and the destination process the active packets (Figure 1) using a specific code.

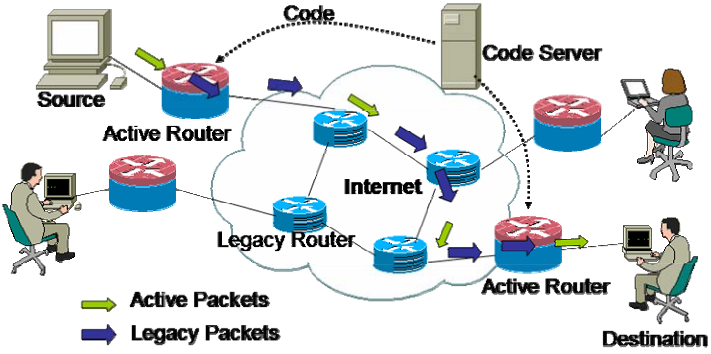


Fig. 1. Scenario of a multiservice network based on a programmable network

In some programmable networks, users can introduce their own executable codes into the active routers, but this is not a pragmatic solution because of the risk of introducing malicious codes. So, in order to allow the network administrator to control the codes running in active nodes, we propose to use code servers. Every active packet carries the code identifier, which identifies the code that the active routers must execute to process the active packet itself. When an active router receives an active packet, if it does not have the code to process it yet, it will download the code from a code server.

Active routers consume resources when the multiservice network offers the services demanded by the users. We will define a service model that forces the users to request a service before using it, so the multiservice network can accept or refuse the service according to the available resources. These services must be controlled in order to offer just the authorized services. So, active routers must only process the active packets that belong to an authorized service (e.g. transcodig service).

There are some approaches towards a programmable network in which active routers need to know who their closest active routers are in order to send them the active packets, while in other approaches active routers that process active packets do not need to know this information (its IP address). In this case active packets are sent to the destination and intercepted by the active routers in the path. We suppose that in a generic scenario of programmable networks, the active routers do not need to know the topology (the other active routers). This supposition allows us to propose a generic security solution valid for both programmable network technology approaches. In addition, it is a pragmatic requirement that the end systems do not need to know the active routers (its IP address) in order to send them active packets.

A programmable network could experience changes in topology which can be produced by changes in the network routes by new active routers that appear in the network or when an active router is down. The changes in topology can cause the changes to take place suddenly, as new active routers start to process the active packets of a service, or that other active routers suspend the processing of active packets. The security architecture must be immune to changes of topology.

It is assumed by the scientific community that the active routers will be located on the edge of the network, where a higher processing power to packet throughput ratio is possible. So, we consider that active routers will be located in the ISP networks that are on the edge of the Internet, which offer services directly to the users.

We propose to use SARA [4, 5] as active router because it follows some of the aforementioned ideas: (1) SARA uses a code server to download codes executed by the active routers, in this way codes can be controlled by network providers. (2) The SARA architecture allows upgrading legacy high-speed routers to work as active routers by delegating active processing to an external entity called assistant. (3) Active packets sent by SARA, have set the router alert [6] IP option. These active packets are sent between two end systems (Source and Destination), using the traditional IP routing. The router alert option allows the active routers in the path to catch active packets, process them and finally queue them in the router output to follow the journey towards their destination. This avoids costly tunneling management. (4) Active routers can be configured dynamically to pick up legacy IP packets compliant with a predefined pattern. The configuration of the pattern is carried out via active packets acting as control packets. The pattern can be specified using fields within the packet headers (e.g. source and/or destination IP address, transport protocol, source and/or destination ports…). The set of packets that match a specific pattern are called a flow.

3 Security Problem

Active routers must be protected against security attacks. An active router is more vulnerable than a legacy router because a active router processes external codes and active packets that may change its own behavior. The deployment of an active network must be carried out by authorized users, so active packets sent by these users, which allow the programmable network to be configured, must be authorized and authenticated. An efficient security mechanism of authorization and authentication has to be used so as not to consume too many resources of active routers. Code executed in active routers have to be downloaded securely from trusted code servers using services that guarantee authentication and confidentiality of codes.

In [3] we propose a security architecture to protect programmable networks such as SARA from malicious active packets and codes. This security architecture allows users to obtain authorization to send active packets from a specific source to a destination. So a user is authorized to obtain a certain service by sending active packets. Active packets and code are protected using cryptography.

Once active packets and code are protected, we focus attention on legacy packets, which are picked them up and processed by active routers. As we stated before, once a pattern has been configured, a predefined flow of legacy packets, which are sent from a source to a destination, are processed by the active routers in the path (e.g. by the transcoding service); we will call these legacy packets *legitimate packets*. A malicious entity could generate fake legacy packets that fulfill the specified pattern. By injecting these fake legacy packets into the same path as the legitimate ones (see fig. 1), the following active routers in the path will process them. So, these active routers use more resources than predicted. Hence, the injection of fake legacy packets can provoke a Denegation of Service (DoS) at active routers. We will call these fake legacy packets *inserted packets*.

The security goal here is to protect active routers. Cryptographic mechanisms could be used to identify the inserted packets, but we have rejected this option because the protection mechanism should be transparent to end systems applications. That is to say, legacy packets have to arrive to destination in a transparent way, even if the active router near the destination is down, the legacy-packet format must not be modified (e.g. introducing cryptographic information that the destination does not know how to process).

The state of the art in insertion attack detection usually uses mechanisms based on the observation of traffic at a point in the network, and the detection of an abnormal model of traffic behavior [7, 8, 9]. These techniques usually have a certain probability of making an error in the detection because they are based on a probabilistic interpretation of the observation. In this paper we propose a cooperative mechanism based on the observation of traffic at two points in the network. Furthermore, we use the programmable capability of the active routers to deploy this cooperative mechanism as a service. By using cooperative detection it is possible to obtain precise information in most cases, which allow us to reduce the risk of making an error in detection.

4 Protection Against Insertion of Packets

4.1 The Basic Description of Detection

The security mechanisms described in this section are for the detection of attacks based on the insertion of packets in a flow of legitimate packets that must be processed by active routers. This mechanism is applied to each segment of the network which is delimited by two active routers or by an end system with active application support and an active router. So, all the segments that make a path between the end systems can be protected.

At the starting point of each protected segment, a *Signalling Agent* (SA) counts the outgoing packets that belong to the observed flow. At the end of the segment, a *Monitoring Agent* (MA) counts the incoming packets that belong to the same flow. The counter of the packets is sent in a *Signalling Active Packet* (SAP) from the SA to the MA, intercalated among the legitimate flow's packets. So, the MA can verify whether the received counter is equal to the local counter at the moment of receiving the SAP.

A SAP is sent with a certain cadence of legitimate packets. The SAPs carry the current counter of outgoing legitimate packets at the SA and the cadence to send another SAP. The term cadence here refers to the number of packets that the SA has to count before generating the next SAP. If non-legitimate packets are sent during a period of time, a default SAP is sent with the counter at that moment, so a default SAP is sent at least at a predefined frequency. The counter or cadences that are carried by the SAPs may be modified by a malicious entity trying to avoid the detection of an attack. So, to prevent it, signalling packets must been protected against modification or fabrication. Because SAPs are active packets, protection of authentication and authorization mechanisms supported by the active network [3] are valid to protect signalling information.

As an active router would be processing a lot of different flows of legitimate packets when an insertion attack is detected, the active router would prevent more packets from being processed than the previously predefined ones that belong to the attacked flow. So a procedure for discarding packets would be activated as a response against an insertion attack. The discarded packets must be selected from among the following packets that arrive at the active router and that belong to the attacked flow. In the discarding process, is not possible to identify the inserted packets, so any legacy packet (legitimate or inserted) could be selected to be discarded. When an attack is detected, because MA counter is bigger that SA counter, the MA counter is updated with the value of the SA counter.

4.2 Reordering and Duplication and Loss of Packets

Since within IP networks such as the Internet, packets can be reordered, duplicated or lost, we are now going to analyze the influence of these issues on the detection mechanism.

We can see the flow of legacy packets as a sequence of sets of legacy packets, where a set of legacy packets is separated by two consecutive SAPs. Because the network can reorder legitimate legacy packets or SAP packets, reordered packets would change to the previous or following set. As well as reordering, duplications of packets can be provoked because of malfunctions in nodes. This will provoke a false detection of an insertion in the MA.

Figure 2 shows that two legitimate packets of a set are delayed in the next set. In this case, the MA will detect that it lacks two packets when the second SAP arrives. Because the MA counter (C_{MA}) is updated with the value of the SA counter (C_{SA}), when the next incoming SAP arrives, the MA will detect that there are two extra packets in the following set, and a false detection occurs.

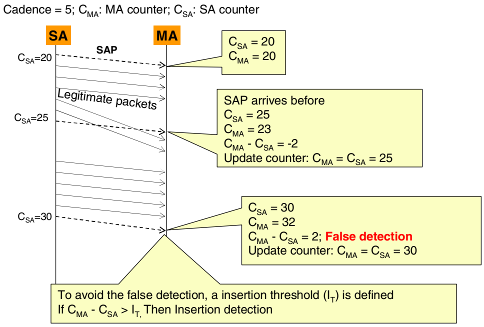

Cadence = 5; C_{MA}: MA counter; C_{SA}: SA counter
SA
MA
SAP
C_{SA}=20
C_{SA}=25
C_{SA}=30
Legitimate packets
C_{SA} = 20
C_{MA} = 20
SAP arrives before
C_{SA} = 25
C_{MA} = 23
C_{MA} - C_{SA} = -2
Update counter: C_{MA} = C_{SA} = 25
C_{SA} = 30
C_{MA} = 32
C_{MA} - C_{SA} = 2; False detection
Update counter: C_{MA} = C_{SA} = 30
To avoid the false detection, a insertion threshold (I_T) is defined
If C_{MA} - C_{SA} > I_T, Then Insertion detection

Fig. 2. Reordering of packets

Tests carried out in [10] show that reordering and duplication rarely happens. So the probability of obtaining a false detection resulting from these events is low. However, to reduce the risk of false detections, we define an *insertion threshold* (I_T) as the difference between the MA's counter (C_{MA}) and the SA's counter (C_{SA}). So, if C_{MA} - C_{SA} is higher than I_T, an insertion attack is detected. On the other hand, the insertion threshold would prevent legitimate packets from being discarded in the event of a small attack which does not mean a significant DoS problem for the active router.

When a packet is reordered, the number of packets skipped in the reordering process is usually small, 3 hops in 87% of reordering cases [11]. So, the minimum value of the insertion threshold should be small (3 or 4 are correct minimum values).

The loss of packets happens at a greater frequency than duplication and reordering. The loss of a legacy packet does not mean a DoS problem for the destination active router. The MA in this active router calculates the difference between the local and the received counters, and because $C_{MA} – C_{SA}$ is less than 0, no detection takes place. In this case C_{MA} is updated with the C_{SA} value. Even if an insertion attack compensates the quantity of packets lost, this does not mean a problem because the active router will not process more packets than the ones predicted.

A main problem happens when an SAP is lost. In this case, the MA does not receive the SAP at the expected time. The observation of this situation could be confused with a strong attack in which the MA receives a lot of inserted packets mixed with legitimate packets. In this case, the MA could assume that a strong attack may be taking place and lot of packets has been intercalated between two consecutive SAPs. These two situations are illustrated in figure 3a. The left-hand illustration shows that two consecutive SAPs have been lost and the MA has counted 11 packets from the last SAP received. The right-hand illustration in figure 3a represents a strong attack in which no SAP has been sent because the SA has only counted 2 outgoing legitimate packets (in both situations we use a cadence of 5). In this case the MA has counted 11 packets from the last SAP received, as there are 9 inserted packets. The MA cannot establish what has really happened until the next SAP arrives. So, in both situations the MA would assume the worst of the situations, which is a *strong insertion attack*.

We will consider that a strong insertion is taking place when a defined number (e.g. 3) of consecutive SAP packets do not arrive when they are expected. When strong insertion is detected, the response mechanism must be aggressive in order to save the active router integrity. An example of an aggressive response consists in discarding all packets that belong to the attacked flow until an SAP arrives and confirms that the attack has finished. But on the other hand, if false detection has taken place (because 3 consecutive SAP's were lost) an aggressive discarding takes place on the legitimate packets when no attack is really happening.

To reduce the impact of the false detection of a strong insertion, a *request for an SAP* packet (RSAP) is sent by the MA to the SA, when an SAP has not arrived as expected. The MA should receive an SAP every time a quantity of packets equal to cadence plus the insertion threshold has arrived (I_T), in the worst case. Every time this does not happen, an RSAP packet is sent to the SA.

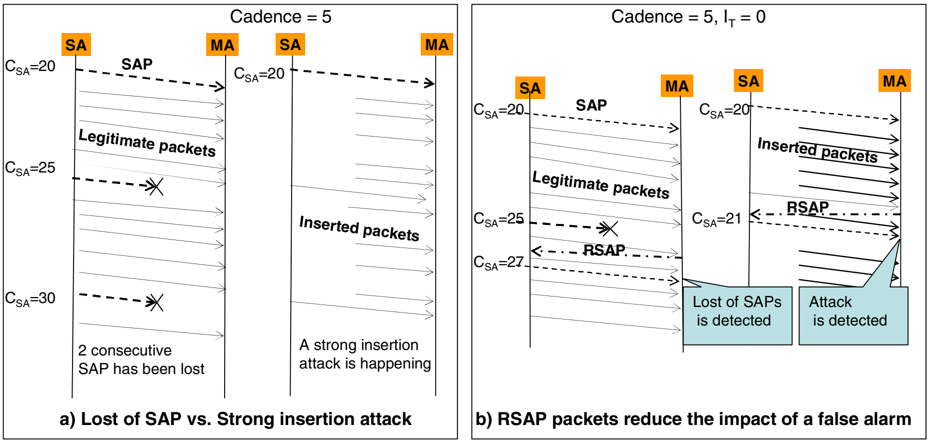


Fig. 3. Reaction against loss of packets

When the SA receives the RSAP, it sends a new SAP with the current value of the counter. So, when a strong attack happens, the SAP packets are sent to the MA more frequently and the attacks are detected beforehand. Figure 3.b illustrates this procedure in both situations, when a strong insertion attack happens and a false strong insertion attack is detected. For simplicity, in figure 3b we have supposed that cadence is 5 and the insertion threshold is 0.

We can see that when the MA has received 6 packets without an SAP, it sends the RSAP. In both cases (left and right), the SA sends an SAP in response to the RSAP received. When the new SAP arrives at the MA, it can be detected whether an attack is taking place or not. The RSAP is an active packet as is the SAP so it has the same authentication and authorization mechanisms that are necessary to avoid unauthorised use.

Summarizing the behaviour of the MA, two events can take place: 1) an SAP arrives and an insertion attack is detected because C_{MA}-C_{SA}>I_T or, 2) an SAP does not arrive when expected. In this case, the MA activates the state of strong insertion attack and sends an RSAP. If the state of strong insertion attack is active, when an SAP arrives at the MA, the following could happen: 1) if C_{MA} - C_{SA} > I_T an insertion attack will be detected, then the MA will change to insertion attack state, or 2) if C_{MA} -$C_{SA} \leq I_T$ a loss of SAP is detected, then the MA will change to normal state.

4.3 Discarding Procedure

We have seen that the MA can produce two kinds of alarm: insertion attack and strong insertion attack. When an insertion attack happens, the MA knows how many packets have been inserted (C_{MA}-C_{SA}), so it can discard the same quantity of packets when the following packets arrive.

When a strong insertion has been detected, the mechanism used to discard legacy packets must protect the active router from DoS, being the least aggressive as possible with legitimate packets. Discarding all legacy packets until being sure that the attack has finished is good for the active router interest but it is an aggressive method on legitimate packets. We have defined a less aggressive *method based on memory*, which consists in discarding all legacy packets in the strong insertion attack state, but when an SAP arrives, it takes into account the amount of discarded legacy packets and discounts it from the amount of legacy packets to be discarded (defined by C_{MA}-C_{SA}). This method is fairer for legitimate packets but it has problems because it produces strong peak in traffic passing through the router (see figure 4).

To find a solution to the peaks appearing in the method based on memory, we propose a discarding *method based on a filter* that consists of measuring the traffic rate when no attack happens. When a strong attack takes place, the MA discards all the packets to maintain the last measured traffic rate. We use a discrete low-pass filter to obtain an estimation of the traffic rate. Some tests have been carried out using different filter coefficients to obtain a good behaviour for discarding when a strong attack takes place. The advantage of discrete filters is that they are easy to implement and the processing cost is low compared to the rest of MA process.

5 Validation Tests

Different tests have been carried out to validate the proposal. At first, the proposed mechanisms were simulated using the simulation tool ns-2 (*Network Simulator v2*), in order to analyze their behaviour in different situations, and then validate them. Then, an implementation was carried out in order to measure the resource consumption and to test its behaviour in a real scenario by tuning parameters as coefficients used for the discharging filter.

The simulation scenario consists of two active routers and a malicious node located between both active routes in order to simulate the loss of packets and insertion attacks. A lot of simulations have been carried out by mixing different situations:

different traffic models, different loss of packet levels, and different attack intensities. The results obtained demonstrate that the mechanism works well in different situations and provokes quite a few false detections, even if the percentage of lost packets is high (2 false detections in 10,000 seconds of observation with a loss of 20%). So the mechanism is resistant to the loss of packets.

The second set of tests tries to verify whether the consumption of resources is proper and scalable. Mechanisms have been implemented for this proposal. The required time for MA and SA to process both the legitimate and SAP packets is 20% less than well-known and the most efficient cryptographic mechanism based on processing packets using HMAC-MD5. Furthermore, we have verified that the processing time increases linearly with the increase in the attack intensity, so the system scales properly.

We have compared the discarding method behaviour using a filter with the discarding method behaviour based on memory. The tests have been carried out on a real Internet scenario using two end points connected to the Internet. The legitimate packets follow a Pareto distribution using an ON/OFF period of 35/15ms and rate of 64 Kbps. We have provoked attacks using CBR traffic of 64 Kbps for 600 seconds. The percentage of loss of packets that the network causes during the test was 6.8%.

In both discarding methods, figure 4 shows the outgoing *non-discarded traffic* compared to the *legitimate traffic* coming into the MA, when an attack is taking place.

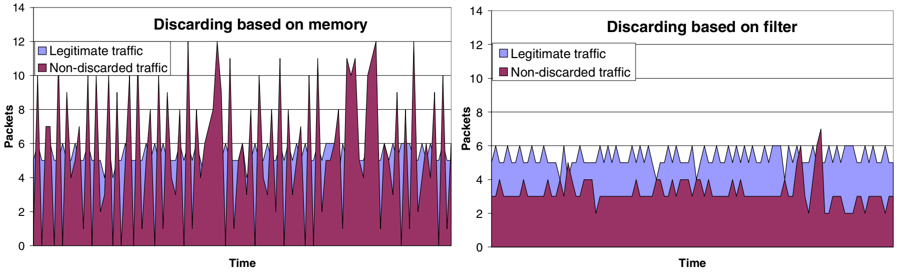


Fig. 4. Outgoing traffic compared to legitimate incoming traffic at the MA

We can see that the non-discarded traffic in the case of discarding based on memory is less stable than the case of discarding based on filters, because a multiple peak of traffic appears that overtakes the incoming legitimate traffic. So, the method based on filters is more conservative to the active router interest, which is the main concern of the security mechanisms. Furthermore, we have seen that discarding based on filters is a little less aggressive to the legitimate traffic than the discarding method based on memory. Finally, the processing time when a packet arrives at the MA increases by 11.8% using filters rather than the memory-based method. We think that is a reasonable cost.

6 Conclusion and Future Work

A model of a multiservice network based on programmable networks has been proposed in this paper. The active routers must be protected against a DoS attack that consists of the insertion of false legacy packets. We have proposed new mechanisms to tackle this problem based on the cooperation between active routers and using the capability of exchanging secure signaling information between active routers. We have carried out tests to validate the mechanisms, to verify the proper consumption of resources, and their scalability. We have proposed mechanisms to discard the packets in case of a strong attack. Finally we can conclude that the proposed mechanism against insertion attacks has a reasonable consumption of resources, and is pragmatic enough to be applied to the scenario described.

For future work we are interested in identifying the legitimate packets in order to carry out a selective discarding. So we are working on marking the outgoing legitimate packets at the SA, using hidden information in active packets to synchronize the SA and the MA.

References

[1] D. Wetherall, U. Legedza, J. Guttag, *Introducing New Internet Services: Why and How,* IEEE Network, Special Issue on Active and Programmable Networks, vol 12, no 3, May/June 1998, pp 12 -19.

[2] D. Larrabeiti, M. Calderón, A. Azcorra, M. Urueña, *A practical approach to Network-based processing*. IEEE 4th International Workshop on Active Middleware Services. 23 de Julio, 2002. Edinburgo, Escocia. TIC2001-1650-C02-01

[3] B. Alarcos, M. Sedano, and M. Calderon, *Multidomain Network Based on Programmable Networks: Security Architecture.* ETRI Journal, vol 27, no 6, pp. 651-665. Dec. 2005.

[4] GCAP: Global Communication Architecture and Protocols for new QoS services over IPv6, IST 1999-10504-GCAP project. http://www.laas.fr/GCAP/

[5] A. Azcorra, M. Calderón, D. Larrabeiti, M. Urueña, *Software Tools for Networking: Simple Active Router Assistant (SARA)*, IEEE Network, Vol 16, N0 4, July 2002.

[6] D. Katz, *IP Router Alert Option*, RFC2113, February 1997.

[7] Thomer M. Gill and Massimiliano Poletto, *MULTOPS: a data-structure for bandwidth attack detection.* In proceedings of the 10th USENIX Security Symposium, August 2001.

[8] Jelena Mirkovic, Gregory Prier, Peter Reiher, *Source-End DDoS Defense.* Second IEEE International Symposium on Network Computing and Applications, April 2003.

[9] Seong Soo Kim, A. L. Narasimha Reddy and Marina Vannucci, *Detecting Traffic Anomalies at the Source through Aggregate Analysis of Packet Header Data*. May 2003. http://dropzone.tamu.edu/techpubs/2003/TAMU-ECE-2003-03.pdf.

[10] Sharad Jaiswal, Gianluca Iannacconne, Chirstophe Diot, Jim Kurose, *Don Towsley, Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone*. In proceedings of the IMW 2002, ACM Press, November 2002.

[11] John C. R. Bennet, Craig Partridge and Nicholas Shectman, *Packet Reordering in not Pathological Network Behavior*, IEEE/ACM Transactions on Networking, Vol. 7, Nº 6, pp. 789-798, December 1999.

Lessons for Autonomic Services from the Design of an Anonymous DoS Protection Overlay

David Ellis and Ian Wakeman

University of Sussex

Abstract. In this paper we report on the design and implementation of a Denial of Service protection overlay, and draw lessons for autonomous services. Our approach is novel in that each node is only aware of a subset of the other nodes within the overlay; the routing topology of the overlay is hidden from internal and external nodes and the overlay uses a distributed monitoring and trust system to detect misbehaving nodes. In meeting these design goals, we have had to move beyond the normal approaches to designing self-configuring and self-monitoring services, and we highlight these issues as being important for the design of future multi-organisation systems.

1 Introduction

Autonomic services attempt to provide services without human intervention. To this end, the services typically are designed so that the various distributed components work together to self-configure and self-regulate in the face of changing demands and resource availability. Nearly all such services follow the following paradigm for their design:

1. Decide on a desirable set of states to be maintained by the system
2. Find a set of local measurements and variables which can be used as indicators of the distance from the desired goal state. These can be solely recovered from local state, or can be found from other machines in the local vicinity.
3. Find a local action which moves the global state closer to the desired state
4. Repeat continuously.

The design problems within a single organisation are identifying the local and global states and actions which allow the service to be built, and which provide at least quasi-stability.

However, when the services are provided by distributed components owned by different organisations, then the design problem becomes much more complicated. Each organisation may be sensitive about revealing information to other organisations. We may therefore have to restrict ourselves to state which can be obtained from solely local sources. Worse, the system must be designed to deal with machines which misbehave, both accidentally and intentionally. We have to design the system to cope with denial of service attacks upon the information exchange and control mechanisms. The system will have to make judgements

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 86–96, 2006.

upon whether information is valid, either by using consensus techniques, or by using policies set up by the system administrators.

In this paper we present the design and implementation of a Denial of Service Protection overlay, similar to SOS [1] and Mayday [2]. However, rather than assuming that each of the nodes within the overlay can be completely trusted, we have aimed to provide a design that can work with untrusted nodes. In this way, we believe the protection overlay can be used across multiple administrative domains and machines safely.

Our design has several novel features, applying design principles suitable for application to sutonomous services between untrusted components:

- Each internal node is only aware of a small subset of other nodes, and the full set of internal nodes is difficult to discover for any attacker.
- The internal routing topology of the overlay is hidden from each of the internal nodes, and from any of the external nodes. This makes it more difficult to launch DoS attacks against key nodes or links within the overlay. Performance is still comparable to other systems.
- The overlay uses a distributed monitoring and trust system to detect which nodes are misbehaving, and subsequently remove them from routing of traffic.

In the following sections, we present the design of the overlay; provide brief descriptions of the construction, routing, choking and trust maintenance protocols for the overlay as appropriate for autonomous services and present results from simulation within NS2 and experience on PlanetLab.

2 Design Overview

DoS protection overlays function through providing a larger set of ingress points for a service, thus providing increased inbound bandwidth, and allowing the actual machine providing service to reject all traffic apart from the known egress points from the overlay. In addition, the overlay should be able to detect the increased traffic from a DoS attack, and react so as to filter this traffic from within the overlay. The key question we have asked ourselves in this work is whether the overlay can be formed from nodes across multiple administrative domains, even to the extent of using the home machines of individual users? Since untrusted nodes can therefore join the overlay, along with machines which are easier to compromise by attackers, we want to design a protection overlay in which the nodes have very limited knowledge about the overlay structure, there is no central control, yet the system still auto-manages. Our basic requirements from the overlay are thus:

- The overlay should reform itself as nodes join and leave the overlay without disrupting connectivity for the end-point services.
- Routing must work without nodes having explicit knowledge of the topology.
- The overlay should identify DoS flows, and filter these flows at the entrance to the overlay.

– The overlay must monitor itself and react to misbehaving member nodes of the overlay.

In terms of the paradigm for autonomous services, each member of the overlay treats the rest of the overlay as a blackbox, making measurements without knowledge of the other members. If nodes may be untrustworthy, then each decision should be independent of other nodes.

2.1 Overlay Construction and Maintenance

Overlay construction must work despite having limited information. We have based our overlay construction algorithm upon the Gossip protocols of Jelasity et al [3]. We have found that such gossip protocols are an excellent match for the composition of autonomous services. They are fast, work well with limited knowledge, and work well to auto-regulate the configuration.

```
Node allNodes, connectedNodes;

function maintainDegree
  if degree == 0 then
    ingress = pick(ingressSet);
    connect(ingress);
  elseif degree < MINDEGREE then
    pullInfo();
  elseif degree > MAXDEGREE then
    n = select(connectedNodes);
    pushInfo(n);

function pushInfo(n)
  nlist = select(allNodes);
  dropConnection(n);
  n.receivePush(nList);

function receivePush(nList)
  foreach n in nList
    allNodes.add(n);
  while(!connect(select(nList)));

function pullInfo
  n = select(allNodes);
  n.getInfo(this,parent(n));

function getInfo(n, referrer)
  if referrer.isValid(n) then
    nlist = select(allNodes);
    n.receivePush(nList);
```

Fig. 1. Gossip pseudo code for graph generation and maintenance

The key requirements of the topologies are:

- The graph should have a high degree of randomness, so that the topology cannot be inferred and extrapolated from a small subset of topological information.
- The graph should be fully-connected with respect to the egress nodes to the end-points.
- The graph should mirror the underlying network so as to reduce the problem of latency stretch.
- There should be a variety of disparate paths between ingress and egress nodes.
- There should be no single points of failure.
- Knowledge of the full topology should be hidden from nodes within the network.

To join the overlay, a node receives the list of public ingress nodes to the network. It then connects to one of the ingress nodes, and executes a pullInfo upon the node. From this node, it selects a node to connect to, and if the connection is successful, this becomes one of the node's outgoing links. The pseudo-code for this is shown in Figure 1.

MINDEGREE is set according to the expected size of the overlay network. We wish to ensure that the overlay graph is fully connected, so following the standard theory of random graphs [4], we need to ensure that for a graph of size N, the average degree of the graph k is $> ln(N)$. We therefore set MINDEGREE to be equal to $ln(N)$. Currently this is set statically, but it should be trivial to allow the degree to be set dynamically according to estimates of overlay size from the ingress nodes.

The public ingress nodes are aware of a large proportion of the nodes within the overlay. However, to have been selected as public ingress nodes, the nodes must have demonstrated themselves to be trustworthy as described below, and we accept the risk of discovering composition from suborning one of the public ingress nodes.

To prevent a node from crawling the network using a sequence of getInfo exchanges, we require each node to also pass across the address of a node which is connected to the queried node. This node can then be used to check the validity of the requester, and to reject the request if too many getInfo requests are being generated.

2.2 Route Learning

Since we wish to prevent dissemination of topology information, routes have to be learnt by experimentation. We therefore have devised a route learning technique based upon the use of probe packets sent out to the end-points within the overlay. An *rlearn* packet is sent out down an outgoing link from an ingress node to probe for a route to a specified end-point. As the packet passes through nodes, these nodes record the passage of the *rlearn* packet. If the *rlearn* packet reaches an egress node responsible for the end-point, then it constructs an *rlearnResponse* packet which is returned along the return route. If a response is received, then we update the rtt statistics to that endpoint out of the corresponding outgoing link and the probability of using that outgoing link for that end-point is increased. We provide the pseudo-code representing the learning algorithm in Figure 2. Rtt statistics are collected and maintained using standard EWMA approaches, as used in TCP. To compile the rtt statistics into the forwarding table probabilities, we calculate probabilities using the following equation

```
function rlearnSend(RlearnPkt p)
  nextNode = pickNode(outgoingNodes);
  rlearnEntry.from =p.src;
  rlearnEntry.timestamp = now;
  rlearnEntry.to = nextNode;
  rtable[p.mark] = rlearnEntry;
  send(p,next);

function rlearnReceive(RLearnRespPkt p)
  discardInvalidPackets();
  r = rtable[p.mark];
  rtt = now - r.timestamp;
  updateRtt(r.service,r.to,rtt);
```

Fig. 2. Pseudo-code for route learning

$$routingtable(i, s) = \frac{rttstore(i, s)}{\sum_{j=0}^{n} rttstore(j, s)}$$

These probabilities are used to bias the choice of outgoing link for a given end-point.

The frequency with which *rlearn* packets are generated is a tunable parameter which affects the rate at which routes are updated and how the network reacts to changes in the underlying topology. We set the frequency of *rlearn* generation based on the round trip times currently experienced by the node.

2.3 Choking and Pushback

Each node monitors the bandwidth used by each incoming link. In the autonomous systems paradigm, this is a local measurement which indicates how far the system is from its goal state. If any incoming link exceeds per-defined thresholds, then the flow is throttled within that node, and a choke request is sent back through the incoming link, the controlling action with the control paradigm. When a choke message is received from an outgoing link, the receiving node reduces its outgoing bandwidth to the sending node. This may cause the receiving node to begin to drop packets, in which case a restriction is sent further upstream. As a side effect the probabilities associated with using that outgoing link will be reduced, as other links with more bandwidth will drop less packets. Nodes may implement per flow restrictions if they wish but the lower granularity is intended to fuel a simpler trust relationship model. Note how the measurement and control action allows nodes to make local policy decisions about what to accept and what to reject. Figure 3 shows the calculation of upstream virtual bandwidths. This algorithm is run periodically, to calculate any upstream bandwidth restrictions. Choke messages are sent upstream only if vbw has changed by some amount, we set this value 0.1. The amount the bandwidth is reduced by is set by another tunable parameter reduceAmount. If this amount is very high, it is equivalent to a complete block from an upstream node.

```
function pushback(VirtualBandwidths, reduceAmount, avgDropped)
  let tbw = temporary array
  foreach upstream node
    VirtualBandwidths[node] =
      reduceAmount * avgDropped[node] / VirtualBandwidths[node]
    if VirtualBandwidths[node] < 0 then
      VirtualBandwidths[node] = 0
```

Fig. 3. Pseudo-code for pushback/choke

According to the rules above, there is no way for a node to re-establish its virtual bandwidth. So an additional thread must be willing to increase bandwidth if the upstream node is now operating within the set thresholds. This algorithm is presented below:

```
function creep(VirtualBandwidths, creepAmount, threshold, avgDropped)
  foreach upstream node
    if(avgDropped[node] < VirtualBandwidths[node] *threshold)
     VirtualBandwidths[node]*=creepAmount
```

Fig. 4. Pseudo-code for pushback/choke creep algorithm

Figure 4 shows the algorithm we used to creep the upstream bandwidths up after they had been reduced. The threshold is needed because we want to allow links with some packet loss increase particularly if the virtual bandwidth is particularly low. How often creep is run is also a tunable parameter and must be carefully balanced against the virtual bandwidth calculations.

2.4 Distributed Monitoring and Trust

Our overlay is built with nodes belonging to multiple organisations and requiring distributed administration (MODA). Since the the different organisations do not necessarily trust one another, there is a need to provide evidential trust measures between the node and the organisations, which can be used to moderate the probabilities of routing through any given node. In particular, if a node is suborned by a malicious third party, we want the system to be self-monitoring to detect anomalous behaviours and isolate the suborned node from the overlay.

We take as our inspiration the Eigentrust work from Kamvar et al [5], in which the transitive trust relationships between nodes are used in the iterative calculation of the left eigenvalue of the normalised trust matrix. The resultant values show the relative levels of trust measure accorded to each node, and can be used to identify suspicious nodes.

In our system, we are trying to detect nodes which either inject traffic into the overlay, or drop traffic unnecessarily. To this end each node records the matrix of incoming (R) and outgoing traffic (S) from its neighbour nodes. These matrices are sent to a central trusted traffic monitor which calculates and monitors the nodes. The traffic monitor combines the received S and R matrices in the following fashion:

$$C = \sum_{i=0}^{N} (S_i' + R_i) + (S_i' + R_i')$$

The global matrix C is normalised over the row totals, and the left hand eigenvalue is calculated by iterative multiplication of C. Note that the Ingress nodes and egress nodes ensure that the matrix is in general irreducible and aperiodic, and the left eigenvalues will converge. Nodes which inject or eject traffic emerge as having values close to one.

Our current implementation uses a centralised node to collect, calculate and monitor the traffic matrices. We are currently working on techniques to fully distribute the trust calculation, in line with the design principles of autonomous services.

3 Simulation Results

We have developed our algorithms within a homegrown simulator, and then have used NS2 to verify our simulation results. In the following simulations, we use topologies of 800 nodes, generated from the Georgia Institute of Technology's topology generator [6]. The overlay has 5 ingress nodes, 40 internal nodes, and 5 egress points to the 2 end-points. There are 10 clients randomly selected which connect to a random ingress point and continually send traffic to the end-point using TCP. For reasons of space, we show only how the total throughput is protected by the overlay. For more complete results from simulation, see [7]. In the following graphs, the throughput is the total bytes received, whilst the RTTs are measured in milliseconds.

Figure 5 shows shows how the overlay deals with a DoS attack from one ingress node in the absence of pushback and choking. When a flood occurs, the bytes received are flattened, and an effective DOS attack has been accomplished. When there is no flood about 600×10^3 bytes get transferred. The graph in figure 6 shows the same scenario but with pushback/choke enabled. We used three settings

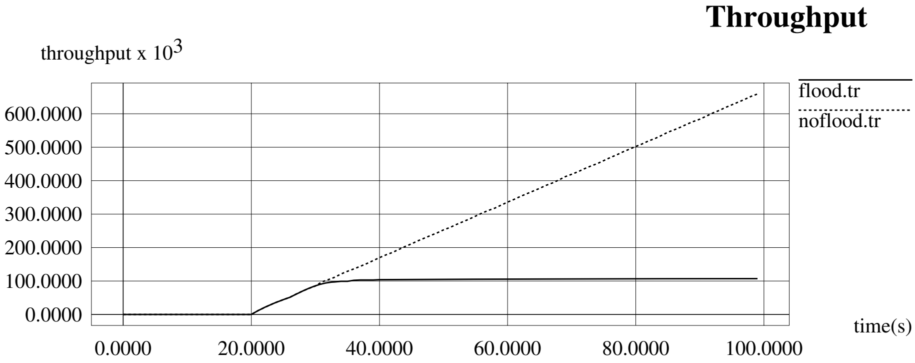


Fig. 5. The effect of a DDOS attack against the network when no pushback is used

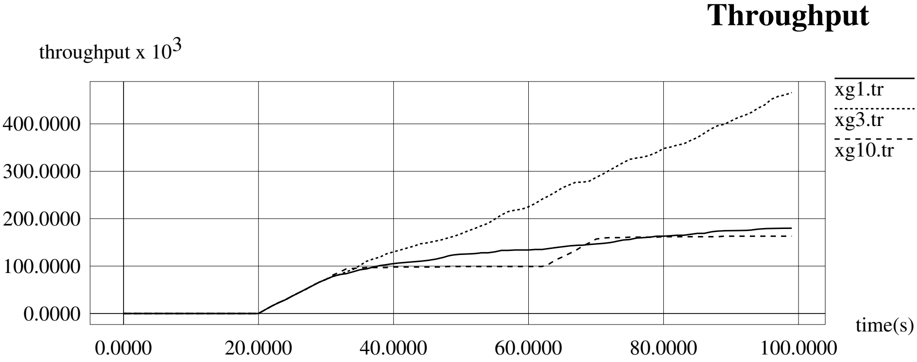


Fig. 6. Using pushback with various creep settings to mitigate a DDOS attack

indicating how quickly the algorithm attempts to "creep" (see section 2.3). The values were 1 second (xg1.tr); 3 seconds (xg3.tr) and 10 seconds (xg10.tr). We can see that when we creep every three seconds we restore the clients throughput to two thirds of what it was when no attack occurred. You can also see that creeping too frequently or not frequently enough hinders the recovery process. This is because creeping too frequently will restore the flooders bandwidth too quickly; and creeping too infrequently will not allow legitimate clients to recover from pushback requests.

4 Implementation on PlanetLab

Many systems perform well within simulation, but fail when deployed on the Internet. By applying the the principles of autonomous services design, we believe that our system will be robust to the various problems that are created within the Internet. We have therefore deployed our code within PlanetLab as a Perl

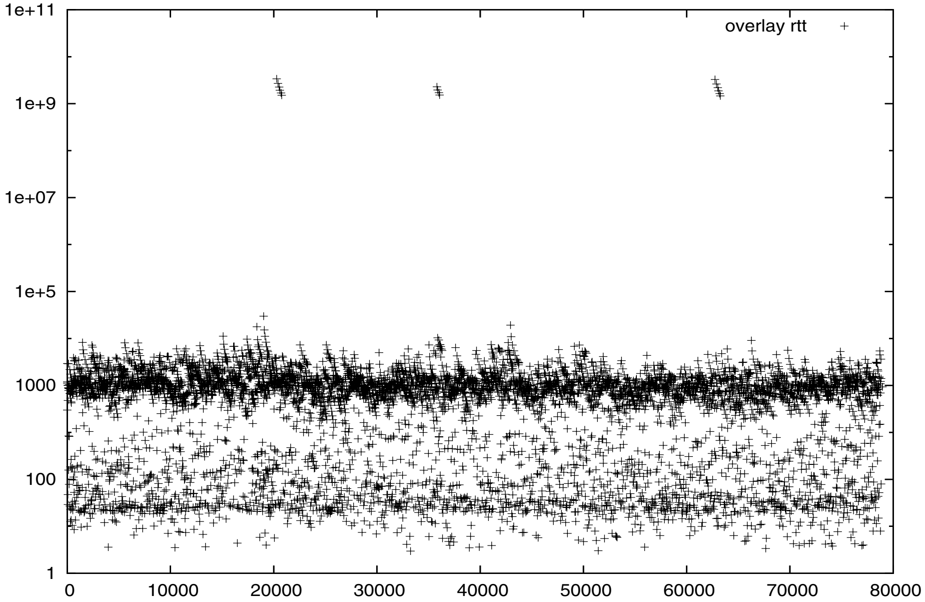


Fig. 7. Round trip time for overlay measurements in milliseconds

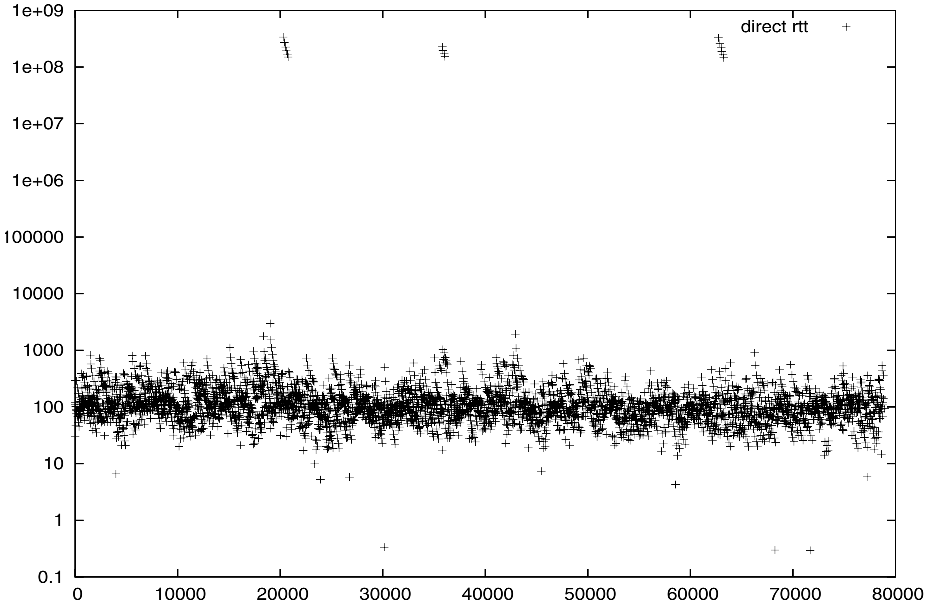


Fig. 8. Round trip time for direct measurements in milliseconds

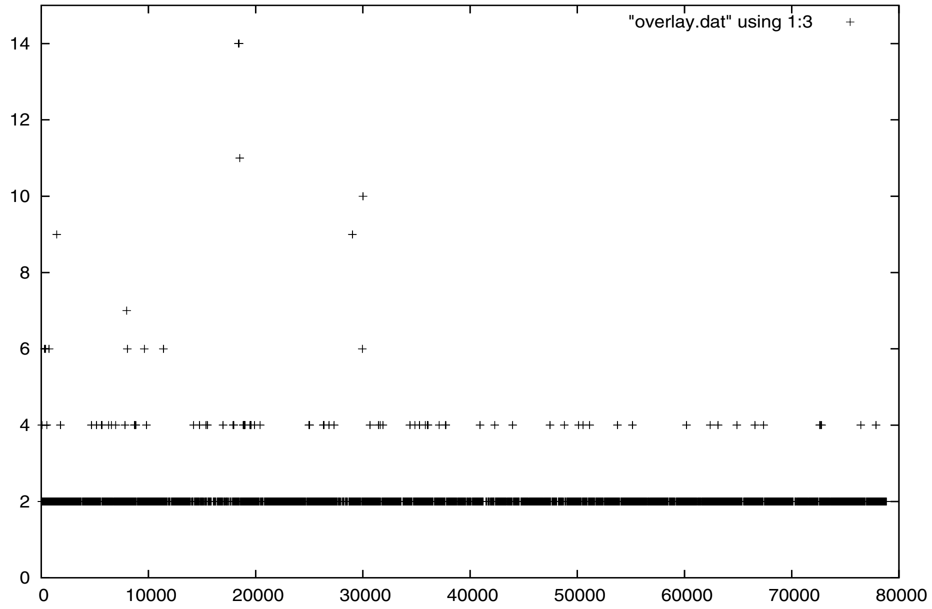


Fig. 9. Number of hops across the overlay

implementation. To pass between nodes, the end-points are included as the Satnet IP packet option, and we use standard NAT translation techniques to transfer the packets within the overlay. The Satnet IP option [8] was used originally to pass stream identifiers through nodes which didn't recognize streams. We have re-used this option to carry the end-point identifiers between nodes within the overlay. Although it has been reported that IP options are not carried through many of the Internet routers, the connections between many of the PlanetLab nodes do allow IP options through, and we have successfully rate limited traffic at low bandwidths. Obviously, we have not been able to run full DoS style attacks within the infra-structure, but the results so far have been encouraging.

For the following results, we measured the performance of the overlay over a 24 hour period on PlanetLab. We used 80 nodes within the overlay, with 5 ingress nodes and 5 egress nodes. Every 15 seconds, we measured the round trip time from the ingress nodes to the egress nodes, both through the overlay, and directly using a ping measurment, and the number of hops. As can be seen, the average latency stretch is very respectable given the problems of context switching on Planetlab nodes, and the system reacts well to the vagaries of PlanetLab connectivity.

5 Conclusion

We have presented the design and implementation of an anonymous DoS protection overlay network. Our simulations and experience on PlanetLab show that the approach can effectively reduce the effect of a DoS attack. If nodes can be offered an incentive for participation, such as the use of micro-payments, then such schemes as ours may find a niche for collaborative protection of small to medium scale web sites and services.

We currently do not attempt to maintain packet ordering within a flow. If we were to attempt to pin a route to a flow as soon as we identified a flow, then entire flows may be sent into routing black holes within the overlay, which would be unacceptable. Instead, we are investigating maintaining flow state, and pinning a route to a flow once an acknowledgment has returned.

We have show that the careful design of the overall system to use black box measurements and local actions which can be controlled by local policy can lead to autonomous services between untrusted nodes. We believe that these principles can help build robust services in the future.

References

1. A. Keromytis, V. Misra, and D. Rubenstein. Sos: Secure overlay services. In *SIGCOMM*, Pittsburgh, PA, August 2002.
2. David G. Andersen. Mayday: Distributed filtering for internet services. In *4th Usenix Symposium on Internet Technologies and Systems*, Seattle, Wa, March 2003.
3. Mrk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, August 2005.
4. R. Albert and A.-L. Barabsi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.
5. Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the Twelfth International World Wide Web Conference*, 2003.
6. Ellen W. Zegura, Ken Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE Infocom*, San Francisco, Ca., 1996.
7. David Ellis and Ian Wakeman. Design and implementation of an anonymous dos protection overlay. In *Submitted for publication*, 2006.
8. J.B Postel. Internet protocol. Technical Report RFC791, IETF, September 1981.

An Extensible and Flexible System for Network Anomaly Detection

Thomas Gamer, Marcus Schöller, and Roland Bless

Institut für Telematik
Universität Karlsruhe (TH), Germany

Abstract. Network hazards like attacks or misbehaving nodes are still a great obstacle for network operators. Distributed denial of service attacks and worm propagations do not only affect the attacked nodes but also the network itself by wasting network resources. In wireless ad hoc networks even more hazards exist due to its self-organizing characteristic. A detection of such network hazards as early as possible enables a fast deployment of appropriate countermeasures and thereby significantly improves network operation. Our proposed detection system uses programmable network technology to deploy such a system within the network itself. Doing this without influencing the routing performance seriously demands a resource saving architecture. We therefore propose to use a hierarchical architecture which runs a very small basic stage all the time and loads specialized detection modules on demand to verify the network hazard. In this paper we introduce our system which can detect DDoS attacks, worm propagations, and wormhole attacks.

Keywords: Programmable Networks, Anomaly Detection, DDoS Attacks.

1 Introduction

In today's networks hazards are frequent and comprise various kinds of attacks as well as serious changes of the network itself. To automatically detect such hazards is still a challenge for network operators on the one hand. On the other hand more and more self-managed networks like ad hoc networks evolve. Such networks require an autonomic mechanism to detect hazards and employ fitting countermeasures autonomously. The range of network hazards we have in mind include network attacks like distributed denial-of-service (DDoS) attacks [6,9] or worm propagations [15,11] but also wormhole attacks [8] or misbehaving nodes in wireless ad hoc networks.

The earlier such hazards can be detected the better the network can be protected against them [17]. This requires a detection system within the network. Programmable networks enhance routers to flexibly and dynamically set up new services on that router. Furthermore it eases the update process of service modules since their functionality gets not tightly coupled to the packet forwarding but is loaded on demand. For these reasons we decided to build our system for anomaly detection based on programmable network nodes and to implement service modules which can generate indications of the occurrence of network hazards. If a new kind of hazards has to be detected we just want to add specialized service modules which can detect this new kind of hazards without any changes to the rest of the system.

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 97 – 108, 2006.

With DDoS attacks [6,9] which are a major threatening type the attacker does not exploit a weakness of the victim's operating system or application but aims to overload resources like link capacity or memory by flooding the system with more traffic than it can process. The attack traffic is generated by many slave systems which the attacker has compromised before. The attacker only has to coordinate all these slave systems to start the attack nearly at the same time against a single victim. As soon as the victim is not reachable anymore no reverse traffic is sent back to slave systems or error messages are generated by routers close to the victim. Such changes of the traffic can be detected by combining various anomalies.

Another threat to the Internet today are worms [15,11]. This piece of software automatically exploits security holes in operating systems or applications to infiltrate a system and starts to propagate itself to as many other systems as possible. Today's countermeasures to worms are signature-based detection systems scanning for well-known worms. These systems are typically located at the victim's edge of the internet preventing the worm propagation to a specific network. An earlier detection of such a worm propagation is possible if the detection system is located in the network itself. There a signature-based detection system is not applicable since it needs deep packet inspection which is infeasible without additional special-purpose hardware. Furthermore, a signature-based detection system is not able to detect previously unknown worms at all. To achieve a detection of unknown worms an anomaly-based detection system can be used. Such a system also has to be deployed within the network to ensure an optimal protection of the network. An anomaly-based detection system can collect hints on a worm propagation for example by analyzing the ratio of error messages due to closed ports generated by scanned systems to the total number of connection requests.

In wireless networks other hazards are possible in addition to DDoS attacks and worm propagations due to the different medium type. Since a wireless medium allows mobility and can not guarantee any knowledge about a participating node it is much easier to threaten the routing protocol or certain connections than it is in wired environments. A possible attack in wireless networks is the wormhole attack [8]. By establishing a wormhole an attacker aims at attracting as much traffic as possible to a node controlled by himself. If a proactive routing protocol is used this is achieved by influencing routing metrics in such a way that other nodes assume the attacker in their neighborhood due to the established tunnel. In fact however the attacker is far away. If a reactive routing protocol is used new routes are established through the tunnel since the tunnel enables the attacker to send a route request faster to the destination than this is possible over normal multi-hop communication. If the attacker succeeds in establishing a wormhole he can attack certain connections or the connectivity of great parts of an ad hoc network by dropping packets arbitrarily. Therefore, a wormhole attack causes an anomalous increase of traffic near the tunnel endpoints as well as an increasing drop rate of packets which are routed over the tunnel.

Our approach to build a hazard detection system uses anomaly-based detection functionality, e.g. stochastic anomalies, distribution anomalies or protocol anomalies. All these anomalies give hints to a current hazard. We analyzed various hazards in different scenarios and concluded that a system which can easily be adapted to these scenarios would prove valuable. Such a system for anomaly-based hazard detection is presented in this paper.

The paper is organized as followed: In section 2 we detail on the architecture of the system for anomaly detection and we explain the special characteristics of our system. Furthermore we describe two scenarios which we think the detection system can be deployed in – a small provider network and an ad hoc network. Section 3 presents implementation details for one of the described scenarios, the small provider network, and an evaluation of the system for anomaly detection. Additionally we will go into memory usage details. Section 4 gives a short summary.

1.1 Related Work

There are some existing approaches that try to detect DDoS attacks or worm propagations based on programmable networks. Some of these even use anomaly-based detection mechanisms. Approaches which mainly focus on mitigation and remediation are not discussed here. One approach of Sterne et al. [16] detects stochastic anomalies by using a simple threshold based DDoS detection mechanism. The system consists of three components: the DDoS flood detector, a management station which dispatches an active program in case of a DDoS detection and routers which are active networking nodes and are able to execute the active program. A drawback with this approach is that after detecting a DDoS flood by a sudden increase in packet distribution no further verification of the attack is done but immediately a rate limiter is installed on the active nodes. Another approach, IBAN [4], detects worm propagations based on active networks. Therefore, a management station distributes so called scanners on active networking edge routers. These scanners search for a specific vulnerability on all hosts which are connected to the edge router. This means that only known worms can be detected. For every newly announced vulnerability a new scanner has to be implemented first and distributed in the active network afterwards. If a vulnerable host is detected by such a scanner the management station is informed and the distribution of a blocker to the proper edge router has to be started manually. Thus, this approach uses some kind of signature-based detection and does not react automatically on worm propagations.

The pushback mechanism [10] is activated as soon as congestion occurs on a router. In this case a flooding attack is assumed and the packets which are dropped on the router due to congestion are inspected in more detail. The mechanism supposes that the distribution of dropped packets resembles the distribution of the whole packet stream and rate limits the highest bandwidth aggregate of packets. This is done for further aggregates until congestion has disappeared on the outgoing links. Afterwards rate limiters for the aggregates are installed in upstream active network routers to reduce congestion of the incoming links. This approach has several disadvantages: one of these is the fact that an attack can be detected not until congestion occurs on a router and hence a detection is only possible at the edge of the network. Another problem is the fact that no further verification is done if the rate limited aggregates really belong to an attack. AEGIS [3] proposes the deployment of so called shields which scan for several different anomalies and therefore, have to do a deep packet inspection which can cause a delayed forwarding of packets if no special-purpose hardware is used. Additionally, a commander scans for further anomalies in the aggregated traffic behind the shields and tries to learn fingerprints of normal traffic to detect DDoS attacks. This procedure is likely to use much resources, primarily memory and processor time, and therefore, causes additional costs if new hardware is needed.

Our approach is based on the usage of a software system to achieve network anomaly detection. It is, however, possible to carry out some or all of the proposed detection mechanisms on network processors [13] or some other special-purpose hardware. This causes additional costs and changes to the currently deployed infrastructure but in some cases provides better performance of the system for network anomaly detection. In both approaches – software-based or hardware-based – resource management is needed in order to use the available resources efficiently [7].

2 Architecture and Usage Scenarios

In subsection 2.1 we focus on the architecture of our system for network anomaly detection. Afterwards deployment of the system in two usage scenarios – a small provider network and an ad-hoc network – is introduced in more detail and anomalies usable for anomaly detection are described (see subsections 2.2 and 2.3).

2.1 Architecture

We developed a system for anomaly detection that is hierarchical, anomaly-based, extensible and flexible. A hierarchical system saves resources by splitting anomaly detection, respectively, into several stages. Thus, the common functionality of a node which runs an anomaly detection system is less affected by a hierarchical system than by a system that keeps its whole functionality in one stage. A hierarchical system for example ensures less memory usage on a router since only some easy calculations are needed in the basic stage. A system composed of only one stage has to maintain much more state for anomaly detection and therefore, causes a much higher memory usage. Thus, low resource consumption due to a hierarchical design ensures applicability in different kinds of networks. Because of this characteristic our system can even be deployed on nodes with limited resources like routers in small provider networks or PDAs in wireless ad-hoc networks.

Additionally, it is very easy to introduce new anomalies into the basic stage of the detection system or new aggregate specific anomalies. This flexibility also ensures applicability in different network scenarios. Examples for the definition of aggregates in the basic stage and for detectable anomalies in suspicious aggregates in the second stage are given in this section.

The attack detection system analyzes in its basic stage the packet distribution within specific aggregates and scans for indications of an attack by detecting stochastic anomalies. Therefore, the packet stream is divided on the fly into intervals with a fixed length. Furthermore, aggregates of interest are defined for observation. The notion aggregate in this paper refers to a set of packets with the same characteristics and therefore, predefined aggregates are for example all TCP or all UDP packets in a small provider network or all routing packets in an ad-hoc network. Then for each predefined aggregate the number of packets that belong to this aggregate is counted in every interval. An indication of an attack then is found if the observed number of packets exceeds a predefined packet threshold of the aggregate. To make the system self-adaptable to network load changes a dynamic *packet threshold* representing the average packet count in this aggregate for the last couple of intervals is calculated. At the end of every interval a check for each aggregate is performed if the observed number of packets exceeds the packet threshold. To prevent the system from generating false positive indications and starting the next stages for deeper inspections unnecessarily an *interval threshold* is defined. This interval threshold is necessary due to the self-similarity of internet traffic [12] which can cause normal traffic to exceed the packet threshold even though no attack is currently going on. Therefore, an indication only is generated and further detection stages are loaded if the packet threshold is exceeded in more consecutive intervals than the interval threshold. These detection stages will subsequently check the suspicious aggregate in more detail.

After detecting an indication for an attack or a network problem, respectively, by an exceeding of the threshold a second stage will be loaded. This second stage examines the suspicious aggregate in more detail by scanning for the currently defined aggregate specific anomalies. Additionally the detection system offers the possibility to load several consecutive stages for a more detailed analysis after detecting a stochastic anomaly in the basic stage. Because the suspicious aggregate is only a subset of the whole packet stream more detailed analysis can be done with low resource consumption and thereby low detraction of a node's common functionality.

The advantage of our system for anomaly detection is the fact that it can be used in very different networks and scenarios due to its flexibility. Dependent on the underlying network aggregates can be defined for observation in the basic stage which are likely to show some anomalies if an attack, a misconfiguration or a network problem occurs. Thus, in wired networks DDoS attacks and worm distributions are attack types that are common. Due to this a priori knowledge aggregates are defined based on transport protocols like UDP and TCP or on the network protocol ICMP. In a wireless ad-hoc network further attack types like the wormhole attack or network problems due to a misbehaving node are possible. Thus, in ad-hoc networks additional aggregates can be defined based on different packet types like routing and data packets.

2.2 Attack Detection in a Small Provider Network

The first example for the deployment of our system for anomaly detection is the detection of anomalies which indicate DDoS attacks or worm propagations in small provider networks since these attacks are the most prevalent hazards in this usage scenario. Therefore, the system scans for stochastic anomalies based on a packet and an interval threshold to detect attacks in the basic stage as described above. After detecting a stochastic anomaly in the basic stage the system loads two consecutive stages. The second stage uses a distribution anomaly to make a differentiation between DDoS attacks and worm propagations. This can be achieved by analyzing the distribution of packets into subnet prefixes based on destination addresses. Therefore, the whole address space is divided into subnet prefixes based on the routing table of the node deploying the detection system. If large parts of the suspicious traffic – the number of packets by which the packet threshold was exceeded – are sent into exactly one subnet a DDoS attack is indicated since only one victim is currently attacked. If the suspicious traffic is equally distributed to all existing subnets a worm propagation is assumed since worms spread all over the internet. Dependent on the results of the second stage attack type specific protocol anomalies are scanned for in the third stage to identify either DDoS attacks or worm propagations in more detail. That means that only DDoS specific protocol anomalies are examined in the third stage if the second stage detected a distribution anomaly indicating a DDoS attack. Currently the anomalies used in our system for network anomaly detection offer no possibility to differentiate between DDoS attacks and legitimate traffic with the same characteristics, e.g. flash-crowd events [1]. Some example protocol anomalies that can be used to detect DDoS attacks or worm propagations in the third stage are described first. More protocol anomalies that can be used to detect worm propagations and other DDoS attacks can be found in [14].

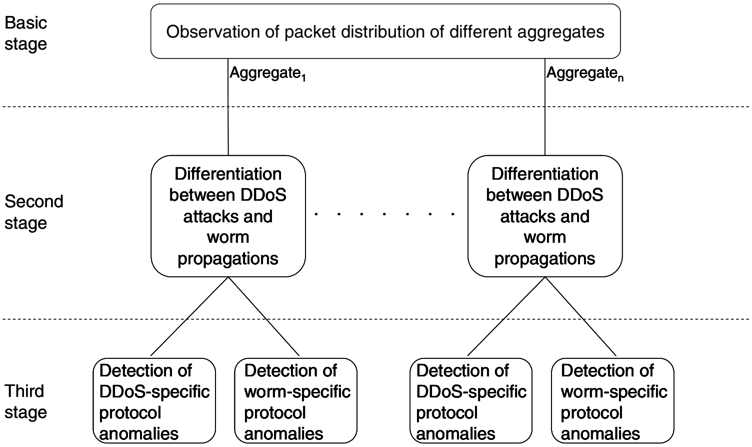

Basic stage
Observation of packet distribution of different aggregates
Aggregate$_1$
Aggregate$_n$
Second stage
Differentiation between DDoS attacks and worm propagations
Differentiation between DDoS attacks and worm propagations
Third stage
Detection of DDoS-specific protocol anomalies
Detection of worm-specific protocol anomalies
Detection of DDoS-specific protocol anomalies
Detection of worm-specific protocol anomalies

Fig. 1. Architecture of system for attack detection in small provider networks

Figure 1 shows the architecture of our detection system as deployed in a small provider network. It consists of three stages: The basic stage scans for stochastic anomalies within the predefined aggregates like *TCP packets*, *UDP packets*, and *ICMP packets*. The further stages are loaded on demand after a stochastic anomaly indicates a suspicious aggregate. The second stage differentiates between DDoS attacks and worm propagations by analyzing the destination address distribution in this aggregate. Depending on this analysis protocol anomaly detection modules for either a DDoS attack or a worm propagation are loaded.

As already mentioned the aggregates *TCP packets*, *UDP packets*, and *ICMP packets* are defined for attack detection in wired networks. The third stage of the detection system is based on the fact that most of the existing DDoS attacks lead to a breach of symmetry between incoming and outgoing packet classes which belong together by protocol definition. A packet class here refers to a set of packets with the same characteristics, for example all TCP packets with SYN flag set. Thus, a DDoS attack can often be detected by a developing asymmetry of packet classes that belong together.

A SYN flooding attack for example tries to exhaust a victim's open connection storage space by flooding the victim with TCP packets with SYN flag set. Due to the mass of connection requests the victim can only respond to a part of all requests by sending TCP packets with SYN and ACK flag set. All remaining requests are dropped and the victim sends no response if storage space is already exhausted and the TCP instance is already down. This leads to an asymmetry between incoming TCP packets with SYN flag set and outgoing TCP packets with SYN and ACK flag set which can be used to detect this kind of DDoS attack.

Another possible DDoS attack is the smurf attack [2], named after its exploit program, which tries to exhaust the victim's bandwidth by flooding the victim with ICMP echo reply messages. Therefore, ICMP echo request messages with forged source addresses are sent to so called reflector systems. These reflector systems in turn response with ICMP echo reply messages to the forged sender address – the address of the victim. If an ICMP request packet is sent to the directed broadcast address of a whole subnet all systems in the subnet will answer the request and thus, amplify the strength of the DDoS attack. This attack leads to an asymmetry between ICMP echo request packets sent by the victim and ICMP echo reply packets received by the victim.

An example for a protocol anomaly that can be used for detection of a worm propagation utilizes the fact that a worm tries to infect other hosts randomly. Additionally, most vulnerabilities a worm tries to exploit are tied to a single port number. A worm propagation based on UDP sends packets with destination port set to the vulnerable port number to randomly selected hosts. If some of these hosts have patched their system already or the vulnerable port is closed anyway these systems send an ICMP packet containing the error message "port unreachable" back to the sender of the UDP packet. If the system or network does not exist at all an ICMP message "host/network unreachable" is generated. Because most worms propagate randomly the ratio of ICMP packets with these error messages will increase during a worm propagation. This protocol anomaly can be used to verify a worm propagation detected already by a stochastic anomaly and a distribution anomaly.

2.3 Attack Detection in an Ad-Hoc Network

The second scenario we looked into are wireless ad-hoc networks. In ad-hoc networks the participating nodes themselves form the networking infrastructure in an ad-hoc fashion to achieve an autonomous, mobile, wireless domain. Our system can be deployed in such a network for anomaly-based detection of attacks, too. In addition to the previous described type of attacks ad hoc specific kinds of attacks can be detected, e.g. wormhole attacks. For this kind of attack an attacker uses a tunnel achieved for example by a directed antenna or by a wired link to send packets over only a few hops to an endpoint far away from the source node. Without the tunnel the packets would have been sent over multiple hops and would have taken significantly more time to reach the endpoint. Due to the tunnel – and thus, the "short" distance between source and destination – the attacker can achieve that much more traffic is sent over the tunnel endpoints controlled by himself than normal routing would do. Thereby, the attacker is able to attack certain connections or the connectivity of a great part of the network by dropping packets that are routed over the tunnel controlled by himself. Figure 2 shows the architecture of our anomaly-based attack detection system in wireless ad-hoc networks.

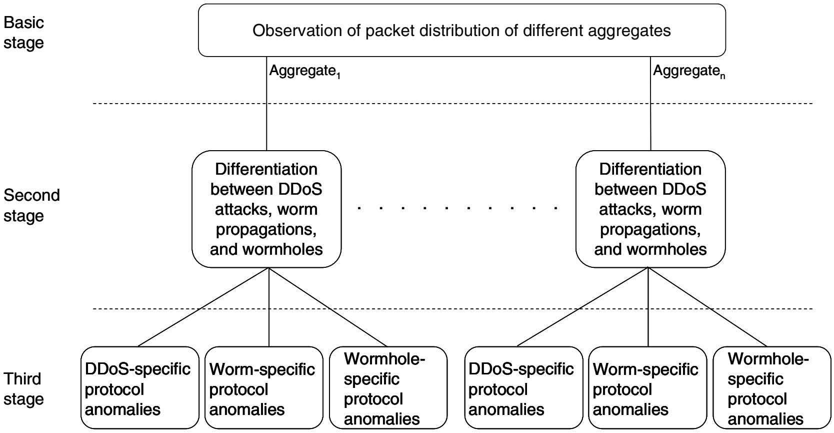

Basic stage
Observation of packet distribution of different aggregates
Aggregate$_1$
Aggregate$_n$
Second stage
Differentiation between DDoS attacks, worm propagations, and wormholes
Differentiation between DDoS attacks, worm propagations, and wormholes
Third stage
DDoS-specific protocol anomalies
Worm-specific protocol anomalies
Wormhole-specific protocol anomalies
DDoS-specific protocol anomalies
Worm-specific protocol anomalies
Wormhole-specific protocol anomalies

Fig. 2. Architecture of system for attack detection in ad-hoc networks

In case of an ongoing wormhole attack an increase in the number of data packets can be detected in the direct environment of the tunnel endpoints. This is a stochastic anomaly that can be detected by the basic stage. The subsequently loaded next stage analyzes the packets in more detail to distinguish the wormhole attack from DDoS attacks or worm propagations. The differentiation between DDoS attacks and worm propagations can be done by detection of a distribution anomaly described already in subsection 2.2 but with a slight difference: the address space is divided into host addresses instead of subnets due to the lower number of nodes in wireless ad-hoc networks. The differentiation between these two attacks and wormhole attacks can be done by another distribution anomaly. During an ongoing wormhole attack the distribution of next hop addresses in the forwarding table tends towards one address that is the next hop for almost all known destination addresses. To verify a suspected wormhole attack the third stage analyzes the suspicious aggregates and checks if the packet drop rate exceeds a certain threshold. This can be realized by a protocol anomaly module which scans for asymmetries in number of data packets to number of acknowledgement packets ratio for instance. Therefore, the third stage gets enhanced with this mechanism if deployed in a wireless ad-hoc network. The verification of worm propagations or DDoS attacks is done as described in the small provider scenario.

3 Implementation and Evaluation

The described attack detection system has been implemented on the programmable network platform FlexiNet [5] based on the Linux operating system. Thus, a service module can install iptables filter rules according to the PSAMP packet selection definition [18] which allows packet selection based on packet header field matching as well as packet sampling schemes. A selected packet is fed to the FlexiNet execution environment via the netlink interface. To preserve the packet order of all flows a copy of the selected packets is fed to the execution environment instead of the real packet which is forwarded regularly on the IP layer.

The basic stage of the anomaly detection system is implemented as a service module which is the only module loaded at system startup. This module processes the packets selected by the installed iptables filter rules. If the basic stage detects a stochastic anomaly in any aggregate by an exceeding of the packet threshold in more consecutive intervals than the interval threshold, specialized modules for stage two are loaded.

The basic stage monitors the number of packets within the predefined aggregates. For the simulation of the ICMP echo reply flooding attack we used a background traffic with an average data rate of about 3 Mbit/s. Due to the rather low bandwidth of the background traffic the following simulation and evaluation is only a first step towards a small provider scenario but nevertheless, we can show that the mechanisms of our detection system work. The average ICMP traffic within this background traffic was 1,8 kbit/s. The simulated attack traffic generated about 400 ICMP packets per interval resulting in an attack data rate of about 3,6 kbit/s of additional ICMP traffic. That is 0,12 % of the overall traffic.

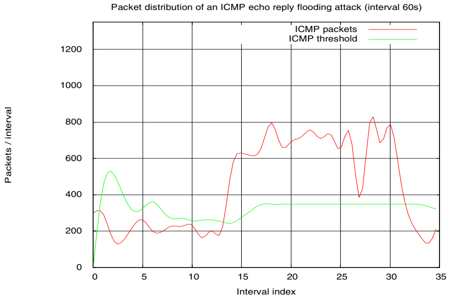
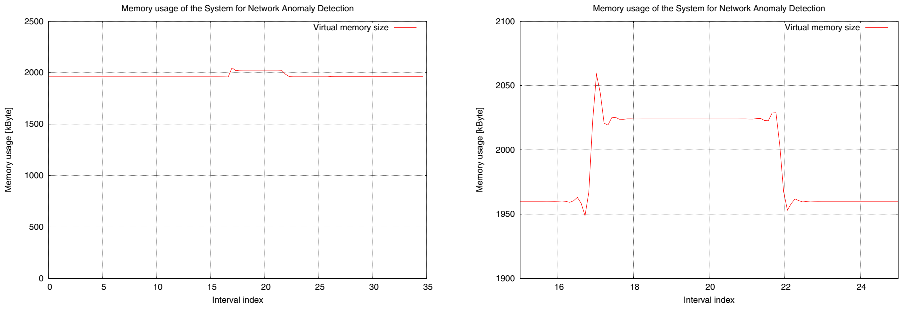


Fig. 3. Packet distribution of an ICMP echo reply flooding attack

The combined traffic – background and attack traffic – was analyzed by our attack detection system. This packet stream is shown in figure 3. Additionally the packet threshold which is used to detect attacks in the ICMP aggregate by a stochastic anomaly is shown. The red line shows the observed number of ICMP packets per interval whereas the green line shows the packet threshold. The threshold is calculated as described in section 2.1 based on the aggregate's average packet count for the last couple of intervals. If an indication is generated due to an exceeding of the threshold for more consecutive intervals than the interval threshold, the threshold remains constant while the attack is running. We can clearly see that the simulated attack begins in interval 13 since the basic stage detected an indication for an attack in this interval. The exceeding of the packet threshold in more consecutive intervals than the interval threshold in the aggregate *ICMP packets* results in loading further stages of the detection system in interval 17. The second stage checks only the suspicious aggregate *ICMP packets* for a distribution anomaly which provides a differentiation between a DDoS attack and a worm propagation. In the simulation one specific subnet could be detected which most of the traffic is sent to by analyzing the distribution of suspicious packets to subnets. Thus, the third stage is loaded that checks only those packets of the suspicious aggregate that are sent into the suspicious subnet for DDoS specific protocol anomalies. In our simulation the third stage was able to detect an asymmetry between incoming ICMP echo reply packets and outgoing ICMP echo request packets as described in subsection 2.2.



(a) Entire anomaly detection

(b) A cutout

Fig. 4. Memory usage of the framework during an ongoing attack detection

Besides validating the functionality of our approach with this setup we also measured the resource consumption of our system. Figure 4a first shows the memory usage of the detection system during the ongoing attack detection described above. The graph shows the system's virtual memory size which is composed of code size, heap size and stack size. The lion's share of the memory usage of about 1750 kBytes is consumed by the execution environment itself. After loading the basic stage about 1960 kBytes virtual memory are used. In interval 17 memory usage increases due to the loading of the further stages. Figure 4b again shows the memory usage of the system during the attack detection phase but only intervals 15 through 25 with an adjusted y-axis scale. During this phase about 100 kBytes additional memory are allocated for the second stage. The third stage even needs just about 70 kBytes additional memory. After unloading the third stage in interval 22 the memory usage again is 1960 kBytes as it was before detecting a stochastic anomaly. These measurements show that our modular and hierarchical system needs just a small amount of virtual memory.

4 Summary

In this paper we presented a system for network anomaly detection which is hierarchical, anomaly-based, extensible and flexible. These characteristics provide for the ability to be deployed in different network environments as well as the ability to detect various network hazards. DDoS attacks, worm propagations, and wormhole attacks are examples of such network hazards. We showed how stochastic anomalies, distribution anomalies, and protocol anomalies can be used to detect these and introduced the architecture of our detection system in a small provider network scenario and a wireless ad-hoc network scenario.

A simulation of an ICMP echo reply flooding attack shows that our anomaly-based system is able to detect DDoS attacks. Furthermore we verified that our hierarchical and modular system needs only a small amount of memory to realize the detection functionality.

In this paper simulation and evaluation were done only with low-bandwidth background traffic. Thus, future research has to address simulations using background traffic with a higher bandwidth to simulate a more realistic small provider network. In this context, the detection performance – e.g. the number of false positives – has to be examined. Furthermore, some work has to be done to achieve a differentiation between network hazards like DDoS attacks and legitimate traffic with similar characteristics, e.g. flash-crowd events.

References

1. I. Ari, B. Hong, E. Miller, S. Brandt, and D. Long. Managing flash crowds on the internet, 2003.
2. CERT. CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks, 2000.http://www.cert.org/advisories/CA-1998-01.html.
3. E. Y. Chen. Aegis: An active-network-powered defense mechanism against ddos attacks. In *IWAN '01: Proceedings of the IFIP-TC6 Third International Working Conference on Active Networks*, pages 1–15, London, UK, 2001. Springer-Verlag.
4. W. L. Cholter, P. Narasimhan, D. Sterne, R. Balupari, K. Djahandari, A. Mani, and S. Murphy. Iban: Intrusion blocker based on active networks. *dance*, 00:182, 2002.
5. T. Fuhrmann, T. Harbaum, M. Schöller, and M. Zitterbart. AMnet 3.0 source code distribution. Available from http://www.flexinet.de.
6. L. Garber. Denial-of-service attacks rip the internet. *Computer*, 33(4):12–17, 2000.
7. A. Hess, M. Schöller, G. Schäfer, A. Wolisz, and M. Zitterbart. A dynamic and flexible Access Control and Resource Monitoring Mechanism for Active Nodes. June 2002.
8. Y. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks inwireless ad hoc networks. Technical report, Department of Computer Science, Rice University, Dec. 2001.
9. A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks-extended. Technical Report ISI-TR-2003-569b, USC/Information Sciences Institute, June 2003. (Original TR, February 2003, updated June 2003).
10. J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium, Catamaran Resort Hotel San Diego, California 6-8 February 2002*, 1775 Wiehle Ave., Suite 102, Reston, VA 20190, February 2002. The Internet Society.
11. D. Moore, C. Shannon, and K. C. Claffy. Code-red: a case study on the spread and victims of an internet worm. In *Internet Measurement Workshop*, pages 273–284, 2002.
12. K. Park and W.Willinger. Self-similar network traffic: An overview. In *Self-Similar Network Traffic and Performance Evaluation*. Wiley Interscience, 1999.

13. L. Ruf, A. Wagner, K. Farkas, and B. Plattner. A detection and filter system for use against large-scale ddos attacks in the internet backbone. In *Proceedings of the Sixth Annual International Working Conference on Active Networking (IWAN 2004)*, October 2004.
14. S. Schober. Mechanismen zur Erkennung von Distributed-Denial-of-Service-Angriffen in IP-Netzen. Studienarbeit am Institut für Telematik, Universität Karlsruhe (TH), December 2003.
15. C. Shannon and D. Moore. The spread of the witty worm. *IEEE Security and Privacy*, 2(4):46–50, 2004.
16. D. Sterne, K. Djahandari, R. Balupari, W. L. Cholter, B. Babson, B. Wilson, P. Narasimhan, and A. Purtell. Active network based ddos defense. *dance*, 00:193, 2002.
17. L. Xie, P. Smith, M. Banfield, H. Leopold, J. Sterbenz, and D. Hutchison. Towards resilient networks using programmable networking technologies. In *Proceedings of IFIP IWAN 2005*, Nov 2005.
18. T. Zseby, M. Molina, F. Raspall, and N. G. Duffield. Sampling and filtering techniques for ip packet selection. Internet Draft, draft-ietf-psamp-sample-tech-06.txt, Work in Progress, Internet Engineering Task Force, February 2005.

Design and Implementation of a Service Provisioning Platform Using Smart Cards

Vincent Guyot[1] and Nadia Boukhatem[2]

[1] Pierre & Marie Curie University
LIP6 - UMR 7606 CNRS
8 rue du Capitaine Scott
75015 Paris - France
vincent.guyot@lip6.fr
[2] GET-Télécom Paris/ENST
LTCI - UMR 5141 CNRS
46 rue Barrault
75013 Paris - France
nadia.boukhatem@enst.fr

Abstract. After having used multimedia services from his desktop computer, the mobile user wishes today to access these services whatever the terminal or network used.

In this paper, we propose an autonomic service provisioning solution to mobile users and suggest the use of a smart card as an autonomy enabler.

Service portability, adaptation and session mobility management is handled entirely by the smart card.

1 Introduction

Autonomic networking covers a large spectrum of themes centred around the networking "selfware" concept. The aim is to develop networking systems capable to react to context changes without explicit user interaction. The development process towards these objectives will certainly be evolutionary and gradual. In this paper, we focus on autonomic provisioning of network services and propose adaptable/context-aware service provisioning solutions allowing the mobile users to access their services through heterogeneous terminals and access networks in a seamless way. We suggest the use of a smart card as an autonomy enabler. For the user, service portability, adaptation and session mobility management is handled entirely by the smart card. To access his services, the user has only to insert his smart card whatever the terminal. Our solution is developed as a first step towards a totally autonomic service provisioning and deployment. Indeed, automony here concerns the point of view of the user rather than the network.

The mobile terminals have evolved. From simple telephony mobile phones, they got memory and execution environments to become intelligent terminals.

The recent developments of wireless technologies and mobile access networks, covering large areas and providing increasing rate, allow the mobile user to remains online wherever he is. The evolution of both terminals and access networks

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 109–118, 2006.

enable the development of new services where mobility is of a main concern [1]. In addition, the services have to be personalizable and portable, and have to adapt automatically to the used terminal [2]. When the user switches from one terminal to another, he should be able to resume his session suspended from the previously used terminal [3].

Since fifteen years, the smart card is impossible to circumvent within mobile networks security. The recent technical evolutions within the smart card field enable the smart card to embed much data processing power.

We propose a service provisioning architecture based on the usage of a personal smart card. This architecture has been developed in the context of various research projects[1]. This environment enables the mobility of the session, the personalization and portability of the services, in a secure way.

This paper is organized as follows. Related work is presented in section 2. In section 3, we provide the main requirements of the service provisioning system. The benefits of using a smart card are explained in section 4. We describe the architecture of the service provisioning system in section 5 and give its main characteristics in section 6. Technical details about our implementation are finally provided in section 7.

2 Related Work

Related work has been done in several European research projects, investigating different ways to realize service provisioning.

The CAMELEON project [7] investigated the application of Agent Technology for the purpose of providing service ubiquity. The project developed an agent-based Service Roaming by implementing an Adaptive Profile Manager and a Personal Agent Manager [8], featuring service provisioning.

In [9], Shiaa et al. describe mobility support in TAPAS (Telecommunication Architecture for Plug And Play Systems), a platform and application development environment based on agent technology. The session mobility is handled by agents, both on the local terminal and remote servers, using session information stored in a local server. Bo You proposed an architecture [10] for User Mobility based on a personal smart card. The card contains a unique identifier to authenticate the mobile user. However, this solution relies on remote servers to enable session mobility.

Only few researches have been done on the provision of services to mobile users through smart cards. The CESURE project provided a framework [11] to build adaptable applications using a smart card to store the description of the user's environment. Such information is retrieved from the card to build an application with different parts downloaded from remote servers.

The common view of service provisioning is based on interactions between the terminal and remote service provider servers, where different services, session information and profiles are stored. Our approach is different: we put a part of the

[1] I3[4], MMQoS[5] and Safari[6].

remote server's functionalities within the smart card to enable autonomy. In addition, using a smart card in such a way avoids expensive networking connections and brings benefits to the end user (see section 4).

3 Service Provisioning

The service provisioning platform has been developed considering the following requirements:

Autonomy in service provisioning. To access his services, the mobile has only to insert his smart card on the available terminal. The smart card handles all the operations to provision services automatically and in a seamless way (service portability, adaptation to the current context and terminal capacities, session mobility, etc.).

Service Portability. The mobile user may use different kinds of terminals. He should be able to access his services whatever the terminal used. The aim of service portability is to make the service usable through different kinds of terminals. The behaviour of the service may be modified according to the type of the terminal used, and restricted to a subset of features.

Service Personalization. The mobile user wishes to use his services depending on his own criteria. The service personalization enables to handle preferences previously set by the user. When the services are installed within a given terminal, they are personalized according to the needs of the user. The user can modify his preferences at every moment, within the limits given by his contract.

Session Mobility. When he switches from a terminal to another one, the mobile user should be able to continue to use his services in a seamless way. For this purpose, it is necessary to save the state of the ongoing work environment when the session suspension occurs. This state will have to be restored to enable the session to be resumed on the new terminal. The services have to provide session information describing their working state to enable a future restoration. From the user point of view, his session is mobile.

Security Management. The mobile user accesses his services following an agreement made with his service provider. The user pays to use his services and wishes to use them in a secure way. There is a need to forbid the unauthorized access so that nobody else could use the user services. Moreover, the personal data that the user may provide when he uses his services should remain private. Finally, the service provider wants to control that a given user remains within the limits of his service contract.

Operation and Management cost reduction. The provision of adaptable and personalized services to mobile users introduces complexity and overhead in service management. Solutions have to be found to limit the service management costs and ensure profitability to network/service providers when offering new services to the mobile users.

4 Benefits of Using a Smart Card

We benchmarked several smart cards available on the market [12]. We observed they have been improved significantly improved during the past few years. Technical evolution permitted the birth of new powerful smart cards. The speeds of calculation and transfer are getting faster and faster. At the same time, the size of embedded memory is rising up. In 70s the memory within smart cards was only few bits, now it is possible to store several mega-bytes in a secure way within the new smart cards. For all these reasons, we think the smart card is a perfect versatile token to enable personalized services to be provided to the mobile user.

We put within the smart card features used to be localized in remote servers (see figure 1). The mobile user has only to insert his personal smart card within a terminal in order to rebuild his personal working environment from the smart card.

Several actions are automatically done when the user insert his smart card within the terminal, and when he removes it:

After having inserted the smart card:

- deployment of the services within the terminal,
- service personalization according to the user's profile,
- resuming the session from embedded session information.

Before removing the smart card:

- session information updating within the smart card,
- removing the previously installed services within the terminal.

This innovative way of realizing service provisioning through the usage of a smart card brings several advantages:

Services Availability. The services of the mobile user are always available. They follow the mobile user, stored within his personal smart card.

Scalability. Retrieving information locally, from the smart card, avoids expensive connections to remote servers. This behaviour gives a good scalability to the system, avoiding remote servers that are bottlenecks.

Session Mobility Enabling. A big challenge is to provide session mobility to the user. Using a smart card gives facility to achieve this goal, by recording session-specific information within the card when terminal switching occurs. When the user inserts his personal smart card within another terminal, the information stored within his card enables him to continue the use of his services in a seamless way.

Tight Billing. A smart card is a secure environment that cannot be modified freely by the user. It enables to pay like a banking card, or through prepaid credits like public phone cards.

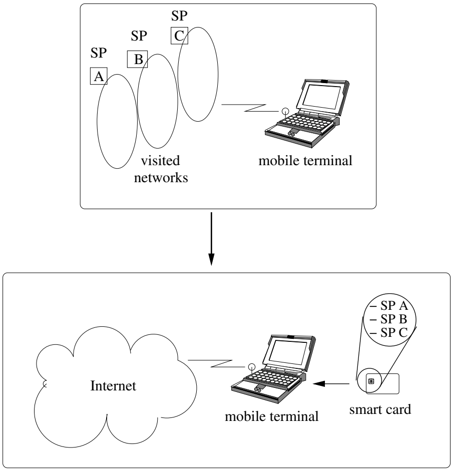


Fig. 1. Service provisioning model evolution

Enhanced Security. The usage of smart cards to access services prohibits unauthorized access. The traditional couple login/password, which remains unsecure, is no longer required to access remote servers. To secure the usage of the smart card, it is possible to use authentication mechanisms, such as a password or biometric methods, preventing to use a stolen card. The network operator knows that the user is not able to modify his subscribed connection parameters, as they are stored within his smart card.

Autonomy. Until now, the mobile user had to configure the terminals that he wanted to use, in order to be able to use his personal services. Using a smart card, this is not the case anymore: his smart card configures his services automatically in order to retrieve his personal working environment.

5 The Architecture

We divided our architecture into three parts [13] to achieve its objectives (see figure 2). These parts are located in different components of the system:

- the CU (Card Unit) is within the smart card,
- the TU (Terminal Unit) is within the terminal,
- the SP (Service Portal) is within the network.

5.1 The CardUnit

The Card Unit (CU) is the personal component of the platform, stored within the smart card. The user carries it wherever he is.

The CU handles several tasks:

- Authentication of the smart card holder and of the terminal used,
- Handling of the user's profile,

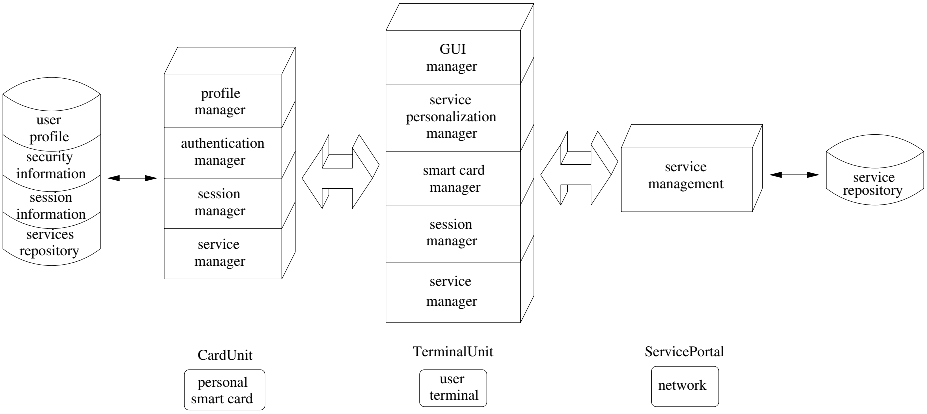


Fig. 2. Architecture components

- Recording and Retrieving the session information,
- Providing the user services.

5.2 The TerminalUnit

The Terminal Unit (TU) is the generic part of the platform. It is built in every terminal of the network operator. It will be personalized by the smart card.

The TU handles several tasks:

- Provision of a graphical interface to the user,
- Personalization of the services, according to the user profile,
- Authentication of the smart card provider,
- Handling the session mobility (suspend/resume),
- Handling new services available from the network operator.

5.3 The ServicePortal

The ServicePortal (SP) is a part of the platform that is within the network, enabling the use of new services.

The SP handles several tasks:

- Presenting new services from services providers,
- Downloading new services for the mobile user.

6 System Characteristics

The service provisioning system has been developed to allow the mobile user to benefit from service personalization and session mobility.

6.1 Service Personalization

The smart card is a safe place to keep sensitive data. New smart cards embed enough memory to store the preferences and the services of the mobile user. The profile and the services being stored within the smart card, it is no longer required to connect to remote servers to retrieve them. The availability of the user profile and services eases the service personalization step.

The processors embedded within the smart cards are faster and faster each year. The smart cards are now powerful enough to ensure the profile and service management. On demand, the smart card can provide the user's preferences to the terminal and manage his services. The mobile user can also modify his preferences, within the limits authorized by his network/service provider.

The different components enabling service personalization [14] are presented in figure 3.

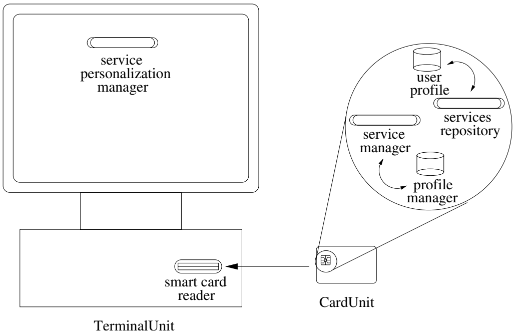


Fig. 3. Components involved in service personalization

- CU's profile manager
  It handles the user profile, which is stored within the card. The profile manager has two different tasks. It replies to the requests from the TU's service personalization manager, sending back the user's preferences stored within the user profile. It also has to update the user profile. The profile manager is the only entity able to modify the profile stored within the card.
- CU's services manager
  This service manager provides the user's services which are stored within the services repository to the TU's service personalization manager. Also, when the user subscribes new services, the service manager stores them within the service repository.
- TU's service personalization manager
  This program has to provide the user with his personal services. It communicates with the CU's profile and service managers to load and configure the services according to the user profile.

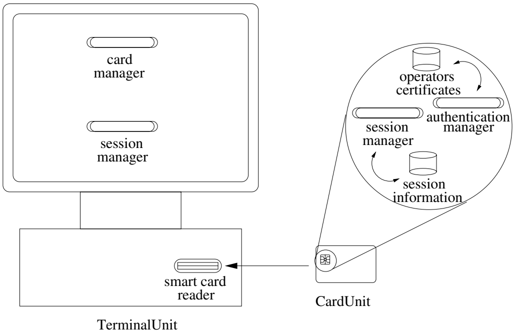


Fig. 4. Components involved in session mobility

6.2 Session Mobility

Different components enable session mobility, both within the smart card and the terminal used. They are presented in figure 4.

- TU's card manager
  Within the terminal, it waits for the smart card being inserted.
  It handles the mutual authentication between the terminal and the smart card.
- CU's authentication manager
  Within the card, the authentication manager participates in the mutual authentication between the smart card and the terminal.
- TU's session manager
  The session manager runs within the terminal. After the smart card is inserted, it obtains the session information from the CU's session manager. It enables the session mobility.
  A graphical interface enables the user to suspend his session and to store the information of the session within his smart card.
- CU's session manager
  Within the smart card, the session manager provides the information to resume a session, and updates the information within the card, after suspending a session.

7 Implementation

We developed a distributed smart card application with both on-card and off-card parts. Our platform has been implemented with portability expectations. Therefore we wrote the off-card part and the services in Java. The on-card part of the platform has been written in JavaCard to be able to use it as-is with different smart cards all embedding JavaCard 2.1 Virtual Machine.

To access the card reader, we used Java OCF standard classes. Since only the card reader PC/SC driver was available, we used a PC/SC to OCF software bridge. Under UNIX systems, we used pcsclite, the PC/SC opensource implementation.

We developed an administration tool to load the personal data within the smart card, such as certificates, the user profiles and small services.

In order to evaluate the capabilities of actual smart cards to handle session mobility, we implemented real services and observed how fast the cards are to suspend and resume a given user session.

These tests are made on several running services: a clock, a stock manager and an email client. The services are all stored within the card and have to be downloaded from the card. Session information to retrieve is composed of the list of the active services, the location of the active windows, the previous display resolution and service-specific session information such as email text or the stock manager remote server address.

We suspended the session. We measured the time between the moment when the system is notified by a click on the appropriate button and the moment when the system notifies that the card can be safely removed. This moment corresponds to the upload of session information within the smart card. Depending on the used smart card[2], the suspension of the session takes 0.8s to 7.1s. Finally, we measured the time to restore the session in a usable manner on the screen. Depending of the used smart card, the restoration of the session takes 4.7s to 51s. The shortest times are obtained by the most recent cards of our testing panel. 1s to suspend and less than 5s to resume a session is fast and very usable in real conditions.

8 Conclusion

During the last few years, technology evolved significantly within the field of smart cards. The processing power increased at the same time than storage capacity and communication capabilities. Smart cards can now enable service provisioning.

This work provides a new solution for service provisioning to the mobile user by using a smart card. Our current implementation runs on laptops and workstations. We expect to provide a version running on PDA-based platforms soon. This next step will ease service provisioning in the real world.

References

1. Y. Li and V. Leung. *Protocol Architecture for universal personal computing*, volume 15 of *IEEE Journal on selected areas in communications.* October 1997.
2. E. Koukoutsis, C. Kossidas, and N. Polydorou. *User Aspects for Mobility.* Acts Guideline SII-G8/0798.
3. G. Forman and J. Zahorjan. *The challenge of mobile computing.* IEEE Computer, 1994.
4. I3 project. http://www-rp.lip6.fr/i3/.
5. MMQoS RNRT project. http://www.mmqos.org/.
6. SAFARI RNRT project. https://safari-rnrt.rd.francetelecom.com/.
7. ACTS CAMELEON project. http://www.comnets.rwth-aachen.de/˜ cameleon/.

[2] Our benchmark panel of smart cards is composed of 30 different Java card models.

8. A. Macaire, D. Carlier, and P. Paradinas. A Personal Agent Manager for Mobile Users. In *EMMSEC*, 1999.
9. Projet TAPAS. http://tapas.item.ntnu.no/.
10. Bo You. Mobile Card Architecture for User Mobility and VHE in Heterogeneous Network Environments. In *Communication Networks & Services Research (CNSR)*, Moncton, Canada, May 2003.
11. CESURE project.
http://www.telecom.gouv.fr/rnrt/rnrt/projets_anglais/cesure.htm.
12. V. Guyot. La Carte à Puce, Vecteur de Mobilité/Smart Card, the Mobility Enabler. PhD thesis, LIP6, 2005. http://rp.lip6.fr/˜ guyot/these-VincentGUYOT.pdf.
13. V. Guyot, N. Boukhatem, and G. Pujolle. Smart Card, the Mobility Enabler. In *5th Workshop on Applications and Services in Wireless Networks (ASWN)*, Paris, France, 2005.
14. V. Guyot, N. Boukhatem, and G. Pujolle. On Smart Card based Service Personalization. In *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Croatia, 2005.

Autonomous Agents for Self-managed MPLS DiffServ-TE Domain

Rana Rahim-Amoud, Leila Merghem-Boulahia, and Dominique Gaiti

Institut Charles DELAUNAY/LM2S, CNRS FRE n° 2848, UTT. 12, rue Marie Curie,
BP 2060, 10 010 TROYES CEDEX, France
{rana.amoud, leila.boulahia, dominique.gaiti}@utt.fr

Abstract. The combination between DiffServ (Differentiated Services) and Multi-Protocol Label Switching (MPLS) presents a very attractive strategy to backbone network service providers. It provides scalable QoS and traffic engineering capabilities. However, the management of such a network is not a simple function and could not be done manually. In fact, it would be much more economic and effective to automatically manage networks. In this paper, we discuss the essential characteristics needed to build an autonomic network. We also propose a novel architecture based on Multi-Agent Systems (MAS) in order to automatically manage an MPLS-DiffServ TE domain. Simulation results are provided to illustrate the efficiency of our proposition.

Keywords: Traffic Engineering, Autonomic MPLS networks management, Multi-Agent Systems.

1 Introduction

In recent years, there has been active research in the field of Multi-Protocol Label Switching (MPLS) and an increasing number of networks are supporting MPLS [3]. One of the most significant applications of MPLS is the traffic engineering (TE) [4]. MPLS-TE enables resource reservation, fault-tolerance and optimization of transmission resources [19]. However, MPLS does not define a new QoS architecture [11] and cannot provide service differentiation by itself. DiffServ (Differentiated Services) [5] defines an architecture for implementing scalable service differentiation in the Internet by defining multiple classes of services. The combination between MPLS and DiffServ allows a differentiation of services and a traffic engineering based on a fast packet switching technology. This network is called MPLS DiffServ-TE.

As networks grow rapidly and traffic conditions change frequently, the management of such a network presents many difficulties and could not be done manually. Therefore, automated management is required to minimize this complexity and to engineer traffic efficiently [9].

In this paper, we discuss the essential characteristics needed to build an autonomic network. We also propose a novel architecture based on Multi-Agent Systems (MAS) in order to automatically manage an MPLS-DiffServ TE domain. A brief description of MPLS and MPLS-DiffServ is presented in the next section. We then study the

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 119 – 131, 2006.

important aspects needed by a network in order to be autonomic. In section 4, we present the MAS as a solution for communication networks. Then, we propose a novel architecture based on a multi-agent system able to automatically manage MPLS DiffServ-TE domains. In section 6, we provide simulation results and evaluate the performance of our proposition. Conclusion and future work are given in section 7.

2 MPLS

MPLS [23] is a technology that uses labels to forward packets by specifying the Forwarding Equivalence Class (FEC). All packets in such a class receive the same treatment in the domain. MPLS domain contains two types of equipments: LER (Label Edge Router) and LSR (Label Switch Router). I-LSR (Ingress LSR) is the LER which puts the label to an incoming packet and E-LSR (Egress LSR) is the one which removes the label from the outgoing packet to return it to its initial nature. An LSR is a high speed router device in the core of the MPLS network. The path between two LERs is unidirectional and is called LSP (Label Switched Path).

2.1 MPLS-TE

Traffic engineering is used to achieve performance objectives such as optimization of network resources and placement of traffic on particular links [19]. In other terms, MPLS traffic engineering routes traffic flows across a network based on the resources the traffic flow requires and the resources available in the network [21].

Current Interior Gateway Protocols (IGPs) always use the shortest path to forward traffic in order to conserve network resources. However, using shortest path is not always the best choice and it may cause the following problems [25]:

1. When different shortest paths from different sources converge at some links causing congestion on those links.
2. The shortest path between a source and a destination is over-used while a longer path between these two routers is under-used.

TE is needed to avoid these problems by optimizing resource utilization and network performance [25].

2.2 MPLS - DiffServ

In MPLS domain, the classification of incoming packets is done just at the entry of the domain by the edge router (I-LSR), by assigning a particular packet to a particular FEC. Within the domain, there is no reclassification and packets are just switched by LSRs according to labels. In DiffServ domain, the traffic classification is also done by edge routers by setting the DSCP (Differentiated Service Code Point) field. In the core network, there is also no reclassification, routers use the DSCP value in the IP header to select a PHB (Per-Hop Behavior) for the packet and provide the appropriate QoS treatment [12].

The functioning of both MPLS and DiffServ consists of 3 main steps: (1) traffic classification, (2) labeling of packets after classifying them, (3) traffic forwarding for

MPLS and routing or switching for DiffServ. In addition, both MPLS and DiffServ are based on aggregation.

The MPLS support of DiffServ is still an open research issue [3]. Currently, there are two solutions [15], the first one is applied to networks that support less than eight PHBs and it uses the 3 Exp (experimental) bits of the MPLS label to determine the PHB. In this cas, LSPs are called E-LSPs. The second solution is applied to networks that support more than eight PHBs. In this solution, the PHB is determined from both the label and the Exp bits and LSPs are called L-LSPs. Each solution has its advantages and its disadvantages and the use of one of them depends on the particular application scenario [19].

In our proposition, we are going to consider the second solution by using different LSPs for different classes of traffic. The effect is that the physical network is divided into multiple virtual networks, one per class. These virtual networks may have different topologies and resources [25]. In this case, three virtual MPLS networks are defined for EF, AF and BE classes. The bandwidth set by administrators on each physical link is partitioned among these MPLS virtual networks. This will provide better resource utilization and each DiffServ level can be treated alone.

We have seen that the TE resolves some serious problems and DiffServ is needed to provide a differentiation of services into the MPLS network. However, an automated management of MPLS DiffServ-TE network is needed to reduce the complexity of the management tasks. The model network should be able to be self-configured, self-managed, self-protected and self-organized. In order to build such a network, some essential characteristics are needed. These characteristics are discussed in the next section.

3 Essential Characteristics to Build an Autonomic Network

In order to build an autonomic network, it is necessary to empower it with some essential characteristics. These characteristics are:

3.1 Decentralization

The current network management scenario is always based on a client-server mode. This centralized management provides a better vision of the global network. However, this represents a heavy and non fault tolerant solution. Therefore, we think there is a real need to decentralize the network control. Control decentralization is obtained by allowing network components to be able to decide on actions to undertake. Furthermore, the component can, if necessary, ask for help from a human administrator or another autonomous component for the realization of some tasks.

3.2 Reactivity

The networks environment is very dynamic and is always in evolution. The router must thus be able to choose the most convenient mechanisms according to the current conditions.

3.3 Proactivity

By being reactive, the router has the possibility of being in phase with the events taking place in its environment. However, we should not rely only on the reactivity to control a router. In fact, a router should envisage the actions to be undertaken.

3.4 Sociability (Cooperation)

The guarantee of end-to-end QoS is a classical problem in networking which is still not totally resolved. Indeed, a given packet may cross several networks, belonging to different operators and thus managed by different strategies. Even if we consider that all these networks are endowed with QoS mechanisms, we cannot guarantee end-to-end QoS. In order to do that, the different networks should cooperate between them and reach agreements to satisfy the requirements of each of them.

3.5 Adaptability (Learning)

In order to realize its goals (accepting more traffic from a given customer, etc.), the router has to carry out some plans to be executed. However, the router must be able to self-evaluate these plans in order to improve its operation. In fact, by observing the results of the plans application, the router can evaluate the relevance of these plans and adapt them, if necessary, to meet more effectively its objectives.

In the next section, we demonstrate that all these characteristics needed to build an autonomic network are offered by the multi-agent solution.

4 Multi-agent Solution

The agents represent a good tool to make networks autonomic. Indeed, a multi-agent system consists of a set of agents which [16] (1) are able to communicate together, (2) possess their own resources, (3) perceive their environment (until a limited degree), (4) have a partial representation of their environment and (5) have a behavior which aims to realize their purposes. Their main characteristics are developed in the following.

4.1 Characteristics of the Agents

The multi-agent systems can constitute a good tool to provide the autonomic scheme by guaranteeing the different characteristics which seem necessary to reach an autonomic behavior. In the following, we demonstrate that all these characteristics are indeed offered by the multi-agent solution:

Decentralization. Multi-agent approach is decentralized by definition. No agent possesses a global vision of the system and the decisions are taken in a totally decentralized way;

Reactivity. One of the basic attributes of an agent is to be situated (situadness, [8]). That is, an agent is a part of an environment. Its decisions are based on what it perceives of this environment and on its current state. This basic characteristic is very important in a context of highly dynamic networks where appropriate decisions must be taken;

Proactivity. An agent can be able to set goals and to realize them by implementing plans, setting up a strategy, starting cooperation with other agents, etc.

Sociability. One of the interesting features of the multi-agent approach is its ability to distribute the intelligence between the different agents composing the system. This implies that an agent can handle some tasks individually but cannot make everything by itself. Many works concerning the concepts of negotiation and cooperation are realized and the research in this field remains very active [24]. The economic theories constituted a good source of inspiration (Contract Net Protocol, auctions, etc.) [13];

Adaptability. In order to provide more flexibility, researchers are interested in the learning in multi-agent systems. A part of the researches is focused on genetic algorithms [7], while the others use the reinforcement learning [14], etc.

After seeing the characteristics provided by MAS which are well suitable to the management of distributed systems, we propose in the next section a solution based on MAS for MPLS networks.

5 Our Proposition

Since the MPLS functioning is based on the use of LSP in order to forward packets, and the MPLS support of DiffServ is also based on the LSP, it seems that the LSP management is the most important need. It includes LSP dimensioning, LSP setup procedure, LSP tear-down procedure, LSP routing, and LSP adaptation for incoming resource requests. In order to effectively control and manage the path of LSPs, one or more attributes can be assigned to each LSP. Such attributes can be Bandwidth, Path attribute, Setup Priority, Holding Priority, Affinity, Adaptability, Resilience, etc. [25].

As agents have to take the convenient decisions into the MPLS domain, so the introduction of these agents will take place into the MPLS decision points. The first step of our research consisted of finding the decision points of the MPLS network which are especially identified on the entry of the domain on the LER routers [22]. An agent will be, as a result, introduced into each LER router in the MPLS domain. In order to control and manage effectively the network and to benefit from the decentralization feature of MAS, we decided to introduce also an agent into each intermediate LSR. Hence, an agent is introduced into each router in the MPLS domain. Actually, each agent is responsible for the router on which it is introduced and for the corresponding interfaces. All these agents form a multi-agent system. These agents interact and communicate together and interact also with the routers and switches in the domain.

The architecture of our agent is shown in Fig.1. The agent includes two entities: the collector entity (CE) and the management entity (ME) which includes, in its turn, two sub entities: the LSP route management entity and the LSP resource management entity. In addition, the architecture contains a Data Base (DB) which is shared between the CE and the ME.

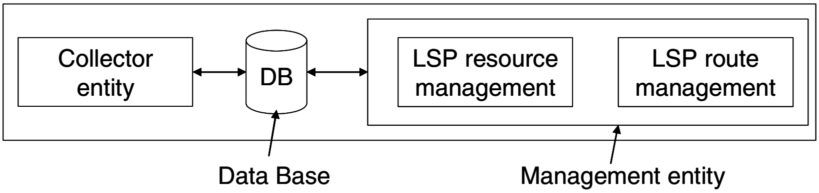


Fig. 1. Our agent architecture

5.1 Collector Entity (CE)

The CEs collect the available bandwidth on physical links and on each LSP. Each CE collects only the information concerning the corresponding router and its interfaces. The CE also collects the network topology information such as the new created LSPs, if an opened LSP is still in use or not, etc. Furthermore, the interaction between agents is done by their CEs by exchanging some of the collected information when necessary. This highlights the social feature of the multi-agent system.

The CE uses SNMP (Simple Network Management Protocol) [10] to collect information from the MIB (Management Information Base) [18] and stores them into the DB. We decided to collect the available bandwidth because we estimate that this is the most important parameter to be treated. It gives us a view of the current network state.

One of the DB tables, called “LSP table”, contains a list of the already created LSPs traversing the corresponding router, their ingress and egress routers, their current available bandwidths and the virtual topologies to which they belong.

5.2 Management Entity (ME)

The ME is an important part of our agent. The ME is responsible for the route and resource management. Precisely, it is responsible for determining when and where an LSP should be created. Indeed, the ME which has access to the DB, uses the stored information to take the appropriate decision. The next step performed by the ME is to automatically implement this decision. The ME contains two sub entities: the LSP route management entity and the LSP resource management entity (Fig. 1).

LSP Route Management Entity. The role of this entity is to route the new LSP on the physical network. More specifically, in case of creating a new LSP, the role of this entity is to decide, for a specific network state, how to select the most suitable route for the LSP avoiding inserting many LSPs on a given link. The idea is to use the routing information generated by a standard IP routing protocol and the local information of the agent in order to reach this purpose. This route will be then used by a signaling protocol such as CR-LDP or RSVP-TE in order to create the LSP.

LSP Resource Management Entity. The role of this entity is to manage the LSP resources. In other terms, this entity has to find the best way to forward the incoming traffic by affecting it to an already created LSP, by re-dimensioning an already created LSP and increasing its allocated resources or by creating a new LSP.

One of the mentioned features of mutli-agent systems is the reactivity. In fact, in order to take the suitable decisions, the multi-agent system has to follow a strategy. In our case, we have proposed a strategy and called it the “LSP creation strategy”. This strategy is described in the next section.

5.3 The LSP Creation Strategy

The general goal of this strategy is to create LSP according to the network conditions. Currently, given the physical topology, the operator has to design a layout or virtual topology by finding an optimal set of paths and a flow distribution over it to accommodate a given demand, as well as to adapt the layout to varying traffic conditions [6].

To design the MPLS layout, there are off-line and on-line proposed approaches. Off-line approaches are based on the estimation of the traffic demand over time. An example of off-line approaches can be the creation of a fully connected MPLS network. This approach consists of creating one or several LSPs between each pair of nodes. This provides a large number of LSPs introducing, as a result, high management cost and complexity. According to Kodialam [17], off-line approaches are not appropriate to MPLS networks due to the high unpredictability of the Internet traffic. Since the off-line approaches present many disadvantages, this solution has to be avoided.

On-line methods calculate paths on demande. Three different on-line approaches can be distinguished: (1) Request-driven, (2) Topology-driven and (3) Traffic-driven.

The request-driven approach is used when MPLS transmits multicast traffic [20]. We do not study this approach because we are not interested in this paper by the multicast case. In the topology-driven approach, a standard IP routing protocol runs and calculates the network’s topology. In addition, a Label Distribution Protocol (LDP) constructs a mesh of labeled paths between ingress and egress LERs according to the routing entry generated by the routing protocol [2]. The constructed path is released only if the corresponding routing entry is deleted. In this approach, LSPs already exist before traffic is transmitted. Thus, a constructed path may not be used because the creation of the LSP was based only on the routing information.

In the Traffic-driven approach, the LSP is created according to the traffic information. When a new request arrives, the corresponding path is established and it is maintained until the session becomes inactive. In this approach, only the required LSPs are setup. This approach conserves labels, bandwidth, signaling and management.

It should be noted that the available bandwidth on a physical link is equal to its maximum bandwidth minus the total bandwidth reserved by LSPs crossing it. It does not depend on the actual amount of available bandwidth on that link [25]. The available bandwidth on a physical link is given by the following equation (1):

$$B_a = B_{rt} - \sum_{i=1}^{n} B_i \ . \tag{1}$$

where B_a is the available bandwidth on the physical link, B_{rt} is its maximum reserved bandwidth, B_i is the bandwidth reserved for the LSP_i and n is the number of LSPs crossing this link.

That means that the establishment of a non used LSP will have bad consequences on the total behavior of the MPLS network. A part of the bandwidth will be reserved without being used. Moreover, another LSP may be prevented from taking a path fault of the lack of the bandwidth. In this context, the traffic-driven technology is more advantageous than the topology-driven one.

The solution, which seems the most logical and the most advantageous to design an MPLS network, is to determine an initial MPLS network topology and to adapt it to the traffic load. Consequently, a topology change will take place when a new LSP is created or released after receiving a real request. Our goal is to decide when to create a new LSP and when to pass a new traffic in an already created LSP. To do that we define the most important factors which can have an influence on the possible decision, these factors are: (1) The requests, (2) The network state and (3) The cost.

A request can be a new bandwidth request, a disabled bandwidth request or a request for tearing-down an existing LSP.

The network state includes the current three virtual topologies such as the created LSP, the existence or not of an LSP between a pair of routers. The network state includes also the LSP attributes (i.e. the available bandwidth, the priority, etc.) and finally, the physical link attributes (i.e. the available bandwidth, the delay, etc.).

The cost includes three different components [3], (1) the signaling cost which is considered only when creating a new LSP or re-dimensioning an LSP. In the other cases, signaling is not needed. (2) The switching cost which depends on the switched bandwidth and the cost defined by the operator. (3) The bandwidth cost which depends on the carried bandwidth and the number of traversed nodes.

Let us discuss the arrival of each type of request. If a request for tearing-down an already created LSP arrives to the entry of the domain. The MAS takes the decision to tear down this LSP and applies it by releasing the corresponding labels and liberating the reserved bandwidth. This information is exchanged between the corresponding agents. As a result, the available bandwidth on the physical link is increased by the value of the liberated bandwidth and all corresponding LSP tables are updated.

Consider now that a request for bandwidth is deactivated. In this case, the MAS takes the corresponding decision and applies it by liberating the reserved bandwidth and increasing the available one of the corresponding LSP. Thus, the available bandwidth on the physical link remains the same and the LSP tables are updated.

Consider that a new bandwidth request arrives between a pair of routers demanding a certain level of QoS. In this case, the first step consists of verifying the existence of an LSP between these two routers in the corresponding virtual topology (EF, AF or BE). This verification is done by consulting the “LSP table”. If an LSP exists, the next step is to compare the available bandwidth of that LSP with the requested one. If the available bandwidth is higher than the requested one, the requested bandwidth is allocated on that LSP and its available bandwidth is reduced accordingly.

If the available bandwidth is lower than the requested one, the multi-agent system verifies the possibility of re-dimensioning the LSP. To do that, the requested bandwidth is compared to the available bandwidth on the physical link (B_a). If the requested bandwidth is lower than or equal to B_a, the multi-agent system decides to increase the capacity of the LSP. In other words, the bandwidth reserved for the LSP in question will be increased by a value equal to the requested bandwidth to be able to forward the new traffic. Consequently, the available bandwidth of the physical link is decreased by the value of the requested bandwidth. If the requested bandwidth is higher than B_a, the multi-agent system eliminates this possibility and verifies the possibility of creating a new LSP on another physical link indicated by the constraint shortest path routing protocol. In this case, the down stream on demand technique is used in order to distribute labels. If no physical link is found, the multi-agent system decides to reject the request.

If there is no LSP between the pair of routers, the multi-agent system reaction will be identical to the one where the requested bandwidth is higher than B_a.

To summarize, our proposition is based on the multi-agent system. Currently, it considers three of the multi-agent systems' features: the decentralization, the reactivity and the sociability. At now, our solution does not consider the proactivity and the adaptability features. These two features will be treated in our future work.

6 Performance Evaluation

In this section, we demonstrate the performance of our proposition by comparing it to the MPLS-TE solution. We assume that in MPLS-TE solution, the constraint shortest path routing algorithm is used for LSP establishment. In addition, both networks used the traffic-driven approach. As our proposition takes into account the bandwidth parameter and it does not consider the preemption priority, we assume that the LSP preemption is not activated for both networks. The objective of the simulation study is to show that the use of our MAS within the MPLS network provides a smaller number of LSPs introducing, as a result, low control traffic, low signaling cost, low management cost and low management complexity. We evaluate the performance of our proposed solution through extensive simulations on the network showed in Fig. 2 using a Java simulator developed in our laboratory. This network includes 7 edge routers and 3 core routers. We assume that the requests arrive one at a time at the network and only one LSP is allowed to be established per LSP request. The source and destination nodes for the requests are randomly chosen from the set of edge routers. The bandwidth request is uniformly distributed between 100 and 200 units. The number of requests varies from 100 to 1000. We set each physical link capacity to 5000 units of bandwidth which is very restrictive in order to oblige a request blocking and thus evaluate the performance of our strategy.

In Fig. 3a, we show the number of LSPs in the network, this number includes the LSPs for the three virtual topologies (EF, AF and BE). We see that when we use the multi-agent system the number of LSPs is lower than that when we do not use the multi-agent system. Thus, by applying our strategy, the number of LSPs is in average reduced by a factor of 2.38 compared to a pure MPLS-TE solution. This improvement in the number of LSPs is due to the re-dimensioning decision taken by our multi-agent system.

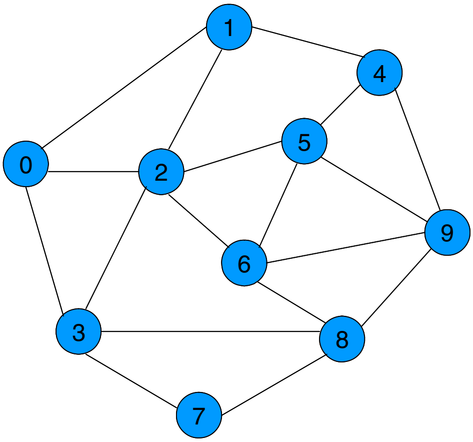


Fig. 2. Network topology

The number of LSPs is not the only performance factor that our proposition enhances. Another performance factor is the number of signaling operations. Fig. 3b. shows the number of signaling operations produced in the network. In fact, our strategy reduces the number of signaling operations by a factor of 1.63. This is attributed to the fact that, in many times, the re-dimensioned LSPs have enough space to forward the incoming traffics and there is no need to create a new LSP. In addition, the simulations show that with the increase of requests number, our proposition provides better results.

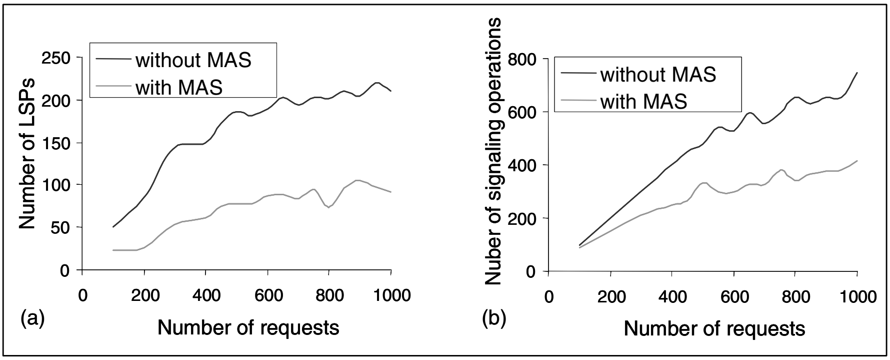


Fig. 3. a) Number of LSPs in the network; b) Number of signaling operations

In order to verify the performance of our proposition, we calculate also the blocking rate. If there is a lack of available capacity to choose the path then the request is rejected. Simulations show that for a number of requests lower than 500 there is no requests blocking in both solutions. Thus in Fig.4, we only plot the blocking rate for requests varying from 500 to 1000 requests. We show that the rate of request blocking is very closed in both solutions. These results mean that our proposition can significantly reduce the number of LSPs and the number of signaling operations. Furthermore, our solution does not degrade the network performance regarding the blocking rate.

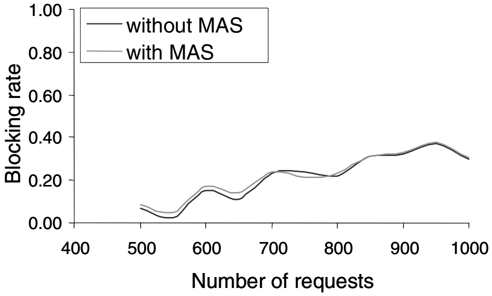


Fig. 4. The blocking rate

7 Conclusion and Future Work

In this paper, we discuss the essential characteristics needed to build an autonomic network. We prove that the multi-agent systems present a good tool in order to reach this purpose. Furthermore, we propose a novel architecture based on the multi-agent systems to automatically manage an MPLS DiffServ-TE domain. Based on the network state, our agents take the appropriate decisions. In our approach, we determine an initial MPLS network topology and then we adapt it to the traffic load. The challenge is to determine when an LSP should be created and when to pass a new traffic in an already created LSP. In order to do that, we propose an LSP creation strategy based on the traffic-driven approach.

Currently, our proposition considers three of the multi-agent systems' features: the decentralization, the reactivity and the sociability. At now, our solution does not consider the proactive and the adaptable features. These two features will be treated in our future work.

Simulation results show that our solution can significantly improve the performance of MPLS by reducing the number of LSPs and the number of signaling operations. Furthermore, our solution does not degrade the network performance regarding the blocking rate.

As future work, we are intended to improve our strategy in order to take into account the proactive and the adaptable features of multi-agent systems. In addition, we will address the possibility of preemption between LSPs. As long term future work, we will define the rules of the route management entity.

Acknowledgment. This is a PhD research supported in part by "Conseil Régional Champage-Ardenne" (district grant) and the European Social Fund. The development is partly realized by two engineer students: Ahmad Sardouk and Wissam Chehadé.

References

1. Aknine, S., Pinson, S., Shakun, M.F.: An Extended Multi-agent Negotiation Protocol. International Journal on Autonomous Agents and Multi-agent Systems, Sycara, K., Wooldridge, M. (eds.), Vol. 8, no. 1 (2004) 5–45
2. Armitage, G.: MPLS: The magic behind the myths. IEEE Communications Magazine, Vol. 38, no. 1 (2000) 124–131
3. Anjali, T., Scoglio, C., de Oliveira, J.C., Akyildiz, I.F., Uhl, G.: Optimal Policy for LSP Setup in MPLS Networks. Comp. Networks, vol. 39, no. 2 (2002) 165–183
4. Awduche, D. O.: MPLS and traffic engineering in IP networks. IEEE Communications Magazine, vol. 37, no.12 (1999) 42–47
5. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: Architecture for Differentiated Services. RFC2475 (1998)
6. Beker, S., Kofman, D., Puech, N.: Off Line MPLS Layout Design and Reconfiguration: Reducing Complexity Under Dynamic Traffic Conditions. International Network Optimization Conference, INOC, Evry, France, N. ISSN: 1762 5734 (2003) 61–66
7. Berger, M., Rosenschein, J.S.: When to Apply the Fifth Commandment: The Effects of Parenting on Genetic and Learning Agents. AAMAS'2004, New York, USA (2004) 19–23
8. Brooks, R. A.: A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, Vol. 2, No. 1 (1985) 14–23
9. Callon, R.: Predictions for the Core of the Network. IEEE Internet Comp., vol. 4, no. 1 (2000) 60–61
10. Case, J., Fedor, M., Schoffstall, M., Davin, J.: A Simple Network Management Protocol (SNMP). RFC 1157 (1990)
11. Cisco Systems, Inc.: Quality of Service for Multi-Protocol Label Switching Networks. Q & A (2001)
12. Cisco Systems: Implementing Quality of Service Policies with DSCP. (2005)
13. David, E., Azulay-Schwartz, R., Kraus, S.: Bidders` strategy for Multi-Attribute sequential English auction with a deadline. AAMAS-2003, Melbourne, Australia (2003)
14. Dutta, P.S., Jennings, N., Moreau, L.: Cooperative Information Sharing to Improve Distributed Learning in Multi-Agent Systems. Journal of Artificial Intelligence Research, Vol. 24 (2005) 407–463
15. Le Faucheur F. et al.: Multi-Protocol Label Switching (MPLS) Support of Differentiated Services. RFC3270 (2002)
16. Ferber, J.: Multi-Agent System: An Introduction to Distributed Artificial Intelligence. Harlow: Addison Wesley Longman (1999)
17. Kodialam, M., Lakshman, T.V.: Minimum interference routing with applications to MPLS traffic engineering. IEEE INFOCOM 2000, Tel Aviv, Israel (2000)
18. McCloghrie, K., Rose, M.: Management Information Base for Network Management of TCP/IP-based internets. (1990)
19. Minei, I.: MPLS DiffServ-aware Traffic Engineering. Juniper Networks (2004)
20. Ooms, D., Sales, B., Livens, W., Acharya, A., Griffoul, F., Ansari, F.: Overview of IP Multicast in a Multi-Protocol Label Switching (MPLS) Environment. RFC 3353 (2002)
21. Osborne, E., Simha, A.: Traffic Engineering with MPLS. Cisco Systems (2003)
22. Rahim-Amoud, R., Merghem-Boulahia, L., Gaiti, D.: Improvement of MPLS Performance by Implementation of a Multi-Agent System. Intelligence in Communication Systems - IntellComm 2005, Springer, Montreal, CANADA (2005) 23–32

23. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC3031 (2001)
24. Sandholm, T. eMediator: A Next Generation Electronic Commerce Server. Computational Intelligence Vol. 18, no. 4, Special issue on Agent Technology for Electronic Commerce (2002) 656–676
25. Xiao, X., Hannan, A., Bailey, B., Ni, L.: Traffic engineering with MPLS in the Internet. IEEE Network Magazine (2000) 28–33

An Efficient Dynamic Bandwidth Allocation Algorithm for Quality of Service Networks*

J. Elias[1], F. Martignon[2], and A. Capone[3]

[1] University of Paris 6, LIP6 Laboratory, 8 rue du Capitaine Scott,
75015, Paris, France
jocelyne.elias@lip6.fr

[2] Department of Management and Information Technology,
University of Bergamo, Italy
fabio.martignon@unibg.it

[3] Department of Electronics and Information, Politecnico di Milano
capone@elet.polimi.it

Abstract. Efficient dynamic resource provisioning algorithms are necessary to the development and automation of Quality of Service (QoS) networks. The main goal of these algorithms is to offer services that satisfy the QoS requirements of individual users while guaranteeing at the same time an efficient utilization of network resources. In this paper we introduce a new service model that provides quantitative per-flow bandwidth guarantees, where users subscribe for a guaranteed rate; moreover, the network periodically individuates unused bandwidth and proposes short-term contracts where extra-bandwidth is allocated and guaranteed exclusively to users who can exploit it to transmit at a rate higher than their subscribed rate. To implement this service model we propose a dynamic provisioning architecture for intra-domain Quality of Service networks. We develop an efficient bandwidth allocation algorithm that takes explicitly into account traffic statistics to increase the users' benefit and the network revenue simultaneously. We demonstrate through simulation in realistic network scenarios that the proposed dynamic provisioning model is superior to static provisioning in providing resource allocation both in terms of total accepted load and network revenue.

Keywords: Dynamic Bandwidth Allocation, Autonomic Networks, Service Model.

1 Introduction

Efficient dynamic resource provisioning mechanisms are necessary to the development and automation of Quality of Service networks. In telecommunication networks, resource allocation is performed mainly in a static way, on time scales on the order of hours to months. However, statically provisioned network resources

* This work is partially supported by the National Council for Scientific Research in Lebanon.

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 132–145, 2006.

can become insufficient or considerably under-utilized if traffic statistics change significantly [1].

Therefore, a key challenge for the deployment of Quality of Service networks is the development of solutions that can dynamically track traffic statistics and allocate network resources efficiently, satisfying the QoS requirements of users while aiming at maximizing, at the same time, resource utilization and network revenue. Recently, dynamic bandwidth allocation has attracted research interest and many algorithms have been proposed in the literature [1,2,3,4,5]. These approaches and related works are discussed in Section 2.

Since dynamic provisioning algorithms are complementary to admission control algorithms [1], in our work we assume the existence of admission control algorithms at the edge of the network that cooperate with our proposed bandwidth allocation algorithm operating inside the network.

In this paper we propose a new service model that provides quantitative per-flow bandwidth guarantees, where users subscribe for a guaranteed transmission rate. Moreover, the network periodically individuates unused bandwidth and proposes short-term contracts where extra-bandwidth is allocated and guaranteed exclusively to users who can better exploit it to transmit at a rate higher than their subscribed rate.

To implement this service model we propose a distributed provisioning architecture composed by core and edge routers; core routers monitor bandwidth availability and periodically report this information to ingress routers using signalling messages like those defined in [2]. Moreover, if persistent congestion is detected, core routers notify immediately ingress routers.

Ingress routers perform a dynamic tracking of the effective number of active connections, as proposed in [6], as well as of their actual sending rate. Based on such information and that communicated by core routers, ingress routers allocate network resources dynamically and efficiently using a modified version of the max-min fair allocation algorithm proposed in [7]. Such allocation is performed taking into account users' profile and willingness to acquire extra-bandwidth based on their bandwidth utility function. The allocation is then enforced by traffic conditioners that perform traffic policing and shaping.

We evaluate by simulation the performance of our proposed bandwidth allocation algorithm in realistic network scenarios. Numerical results show that our architecture allows to achieve better performance than statically provisioned networks both in terms of accepted load and network revenue.

In summary, this paper makes the following contributions: the definition of a new service model and the proposition of a distributed architecture that performs dynamic bandwidth allocation to maximize users utility and network revenue.

The paper is structured as follows: Section 2 discusses related work; Section 3 presents our proposed service model and provisioning architecture; Section 4 describes the proposed dynamic bandwidth allocation algorithm; Section 5 discusses simulation results that show the efficiency of our dynamic resource allocation algorithm compared to a static allocation technique. Finally, Section 6 concludes this work.

2 Related Work

The problem of bandwidth allocation in telecommunication networks has been addressed in many recent works. In [7] a max-min fair allocation algorithm is proposed to allocate bandwidth equally among all connections bottlenecked at the same link. In our work we extend the max-min fair allocation algorithm proposed in [7] to perform a periodical allocation of unused bandwidth to users who expect more than their subscribed rate.

Dynamic bandwidth provisioning in Quality of Service networks has recently attracted a lot of research attention due to its potential to achieve efficient resource utilization while providing the required quality of service to network users [1,2,3,4].

In [1], the authors propose a dynamic core provisioning architecture for differentiated services IP networks. The core provisioning architecture consists of a set of dynamic node and core provisioning algorithms for interior nodes and core networks, respectively. The node provisioning algorithm adopts a self-adaptive mechanism to adjust service weights of weighted fair queuing schedulers at core routers while the core provisioning algorithm reduces edge bandwidth immediately after receiving a Congestion-Alarm signal from a node provisioning module and provides periodic bandwidth re-alignment to establish a modified max-min bandwidth allocation to traffic aggregates.

The work discussed in [1] has similar objectives to our dynamic bandwidth allocation algorithm. However, their service model differs from our proposed model and traffic statistics are not taken into account in the allocation procedure. Moreover, in our work we suggest a distributed architecture implementation, while in these papers only a centralized scheme is considered.

A policy-based architecture is presented in [3], where a measurement-based approach is proposed for dynamic Quality of Service adaptation in DiffServ networks. The proposed architecture is composed of one Policy Decision Point (PDP), a set of Policy Enforcement Points that are installed in ingress routers and bandwidth monitors implemented in core routers. When monitors detect significant changes in available bandwidth they inform the PDP which changes dynamically the policies on in-profile and out-of-profile input traffics based on the current state of the network estimated using the information collected by the monitors. However, this scheme, while achieving dynamic QoS adaptation for multimedia applications, does not take into account the users utility function and their eventual willingness to be charged for transmitting out of profile traffic, thus increasing network revenue.

In [2], a generic pricing structure is presented to characterize the pricing schemes currently used in the Internet, and a dynamic, congestion-sensitive pricing algorithm is introduced to provide an incentive for multimedia applications to adapt their sending rates according to network conditions. As in [2], we take into account users bandwidth utility functions to evaluate our proposed allocation algorithm based on the increased network revenue that is achieved. However, the authors consider a different service model than that proposed in our work and focus mainly on the issue of dynamic pricing to perform rate adaptation based on network conditions.

The idea of measuring dynamically the effective number of active connections as well as their actual sending rate is a well accepted technique [4,6]. In [4], the authors propose an active resource management approach (ARM) for differentiated services environment. The basic concept behind ARM is that by effectively knowing when a client is sending packets and how much of its allocated bandwidth is being used at any given time, the unused bandwidth can be reallocated without loss of service. This concept is in line with our proposed bandwidth allocation algorithm. Differently from our work, however, ARM does not guarantee to the user a minimum subscribed bandwidth throughout the contract duration since unused bandwidth is sent to a pool of available bandwidth and it can be used to admit new connections in the network, in spite of those already admitted.

3 Service Model and Dynamic Provisioning Architecture

We first introduce our proposed service model, then we present a distributed provisioning architecture which implements such service model by performing the dynamic bandwidth allocation algorithm described in Section 4; finally, we present the signalling messages used to assure the interaction between network elements.

3.1 Service Model

We propose a service model that, first, provides a quantitative bandwidth guarantee to users and then exploits the unused bandwidth individuated periodically in the network to propose short-term guaranteed extra-bandwidth. In this process, different weights can be assigned to network users to allocate extra-bandwidth with different priorities; such weights can be set statically off-line, based on the service contract proposed to the user, or can be adapted on-line based, for example, on the user bandwidth utility function.

Our proposed service model is therefore characterized by:

- a quantitative bandwidth guarantee, expressed through the specification of user's subscribed rate;
- short-term guaranteed extra-bandwidth: the network is monitored on-line to individuate unused bandwidth that is allocated with guarantee, during the update interval, to users who can exploit it to transmit extra-traffic;
- a weight that expresses the user's priority in the assignment of extra-bandwidth;
- a bandwidth utility function, $U(x)$, that describes the user's preference for an allocation of x bandwidth units. In line with [8] we consider the utility function as part of the service model. Without loss of generality, we do not consider the pricing component of a bandwidth utility function.

3.2 Architecture and Control Messaging

To implement our service model we assume a distributed architecture constituted by core and edge routers, as shown in Figure 1; traffic monitors are installed on ingress and core routers to perform on-line measurements on the incoming traffic flows and network capacity utilization, respectively.

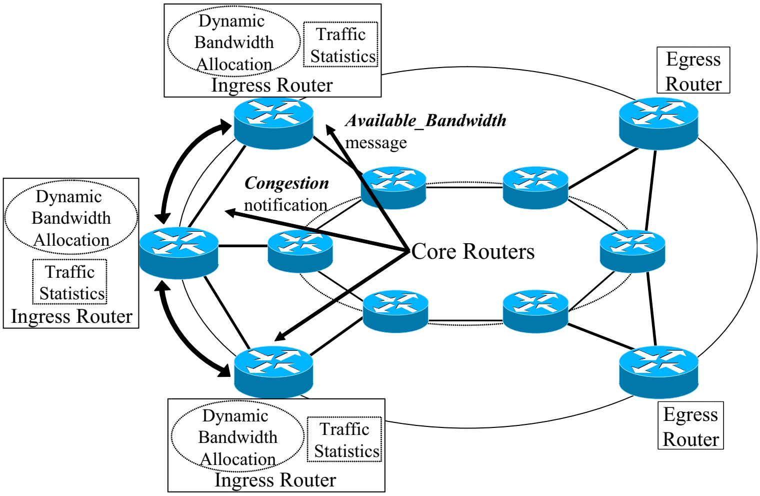

Dynamic Bandwidth Allocation
Traffic Statistics
Ingress Router
Egress Router
Available_Bandwidth message
Congestion notification
Dynamic Bandwidth Allocation
Traffic Statistics
Ingress Router
Core Routers
Dynamic Bandwidth Allocation
Traffic Statistics
Ingress Router
Egress Router

Fig. 1. The proposed distributed architecture that supports dynamic bandwidth allocation

Core routers exchange messages with ingress routers to report the link utilization or to notify a congestion situation. Each ingress router collects the measurements performed by traffic monitors and exchanges periodically update messages with all other ingress routers to report the current incoming traffic statistics. Moreover, a dynamic bandwidth allocation algorithm is implemented in all ingress routers: it takes into account the traffic statistics gathered at ingress routers and the network information reported by core routers to allocate network resources dynamically and efficiently.

The messages exchanged between network routers, illustrated with arrows in Figure 1, are similar to the control messages that have been proposed in [1] to report persistent congestion or resource availability.

4 Dynamic Bandwidth Allocation Algorithm

We propose a novel dynamic provisioning algorithm that allocates network capacity efficiently based on traffic statistics measured on-line. Bandwidth allocation is performed by ingress routers periodically and is enforced using traffic conditioners. We denote the interval between two successive allocations performed by the algorithm as the *update interval*, whose duration is T_u seconds. Moreover, core routers monitor link utilization, and if congestion on some links is detected, bandwidth re-allocation is immediately invoked to solve this situation.

In the following we present in details the bandwidth allocation algorithm, that proceeds in two steps: in the first step, bandwidth is allocated to all active connections trying to match their near-term traffic requirements that are predicted based on statistics collected by ingress routers. In step two, spare bandwidth as well as bandwidth left unused by idle and active connections is individuated on each link. Such available extra-bandwidth is allocated with guarantee during the current update interval exclusively to connections that can take advantage of it since they are already fully exploiting their subscribed rate.

To illustrate the allocation algorithm, let us model the network as a directed graph $G = (N, L)$ where nodes represent routers and directed arcs represent links. Each link $l \in L$ has associated the capacity C_l. A set of K connections is offered to the network. Each connection is represented by the notation (s_k, d_k, sr_k), for $k = 1, \ldots, K$, where s_k, d_k and sr_k represent the connections source node, destination node and the subscribed rate, respectively; furthermore, we assume that each connection has associated r_min_k, which represents the minimum bandwidth the application requires. Let a_k^l be the routing matrix: $a_k^l = 1$ if connection k is routed on link l, $a_k^l = 0$ otherwise. We assume that a communication between a user pair is established by creating a session involving a path that remains fixed throughout the user pair conversation duration. The session path choice method (i.e., the routing algorithm) is not considered in this paper.

At the beginning of the $n - th$ update interval, each ingress router computes the transmission rate, b_k^{n-1}, averaged over the last T_u seconds, for all connections $k \in K$ that access the network through it. This information is then sent to all other ingress routers using control messages as described in the previous Section, so that all ingress routers can share the same information about current traffic statistics and perform simultaneously the same allocation procedure.

The amount of bandwidth allocated to each source k during the $n - th$ update interval, r_k^n, is determined using the two-steps approach described in the following:

- First step: Connections having $b_k^{n-1} < r\_min_k$ are considered *idle*; all other active connections are further classified as *greedy* if they used a fraction greater than γ of their subscribed rate sr_k (i.e. if $b_k^{n-1} > \gamma \cdot sr_k$), otherwise they are classified as $non - greedy$.

  Let us denote by K_i, K_{ng} and K_g the sets of idle, non-greedy and greedy connections, respectively.

  - *Idle* connections are assigned their minimum required transmission rate, i.e. $r_k^n = r\_min_k, \forall k \in K_i$.
  - *Non-greedy* connections are assigned a bandwidth that can accommodate traffic growth in the current update interval while, at the same time, save unused bandwidth that can be re-allocated to other users. Several techniques have been proposed in the literature to predict the near-term

transmission rate of a connection based on past traffic measurements. In this work we only consider the last measured value, b_k^{n-1}, and we propose the following simple bandwidth allocation: $r_k^n = \min\{2 \cdot b_k^{n-1}, sr_k\}, \forall k \in K_{ng}$. In this regard we are currently studying more efficient traffic predictors that could allow improved bandwidth allocation.

- *Greedy* connections are assigned in this step their subscribed rate sr_k, and they also take part to the allocation of extra-bandwidth performed in step two, since they are already exploiting all their subscribed rate.

- Second step: after having performed the allocations described in step one, the algorithm individuates on each link l the residual bandwidth R_l, i.e. the spare bandwidth as well as the bandwidth left unused by idle and non-greedy connections. R_l is hence given by the following expression:

$$R_l = C_l - (\sum_{k \in K_i \cup K_{ng}} r_k^n \cdot a_k^l + \sum_{k \in K_g} sr_k \cdot a_k^l), \forall l \in L \tag{1}$$

where the first summation represents the total bandwidth allocated in step one to idle and non-greedy connections, while the second summation represents the bandwidth allocated to greedy connections.

Such extra-bandwidth is distributed exclusively to greedy connections using the algorithm detailed in Table 1, which is an extension of the allocation algorithm proposed in [7]. This algorithm takes as input the set K_g of greedy connections, the link set L with the residual capacity on each link l, R_l, and the routing matrix a_k^l, and produces as output the amount of extra-bandwidth $f_k^n, k \in K_g$ that is assigned to each greedy connection during the $n-th$ update interval, so that finally $r_k^n = sr_k + f_k^n, \forall k \in K_g$.

Table 1. Pseudo-code specification of the bandwidth allocation algorithm

```
(1) initiate all $f_k^n = 0$, $\forall\ k \in K_g$
(2) remove from the link set $L$ all links $l \in L$ that have
    a number of connections crossing them $n_l$ equal to 0
(3) for every link $l \in L$, calculate $F_l = R_l/n_l$
(4) identify the link $\alpha$ that minimizes $F_\alpha$
    i.e. $\alpha \mid F_\alpha = min_k(F_k)$
(5) set $f_k^n = F_\alpha$, $\forall\ k \in K_\alpha$, where $K_\alpha \subseteq K_g$ is the set
    of greedy connections that cross link $\alpha$
(6) for every link $l$, update the residual capacity and the
    number of crossing greedy connections as follows:

        $R_l = R_l - \sum_{k \in K_\alpha} f_k^n \cdot a_k^l$
        $n_l = n_l - \sum_{k \in K_\alpha} a_k^l$

(7) remove from set $L$ link $\alpha$ and those that have $n_l = 0$
(8) if $L$ is empty, then stop; else go to Step (3)
```

To take into account users weights it is sufficient to substitute n_l in Table 1 with w_l, which is defined as the sum of the weights of all greedy connections that are routed on link l.

It should be clarified that our algorithm can temporarily present some limitations in bandwidth allocation, since the bandwidth allocated to a user can at most double from an update interval to the successive one. This could affect the performance of users that experience steep increases in their transmission rate. In Section 5 we evaluate numerically this effect showing at the same time how it is counterbalanced by increased network revenue in all the considered network scenarios under several traffic load conditions.

5 Numerical Results

In this Section we compare the performance, measured by the average accepted load and network extra-revenue versus the total load offered to the network, of the proposed dynamic bandwidth allocation algorithm with a static provisioning strategy, referring to different network scenarios to cover a wide range of possible environments. Static provisioning allocates to each source k its subscribed rate sr_k.

We are interested in measuring the following performance metrics: the average accepted load and network extra-revenue. The average accepted load is obtained averaging the total load accepted in the network over all the bandwidth update intervals.

We define, in line with [2], the average network extra-revenue as the total charge paid to the network for all the extra-bandwidth utilization, averaged over all the bandwidth update intervals. In this computation we consider only network extra-revenue generated by greedy users that are assigned extra-bandwidth by our proposed dynamic allocation algorithm. Furthermore we assume, in line with [5], that the utilities are additive so that the aggregate utility of rate allocation is given by the sum of the utilities perceived by all network users.

Using the notation introduced in the previous Section, the average network extra-revenue can be obtained averaging over all the update intervals n the quantity:

$$\sum_{k \in K_g} U(b_k^n) - U(sr_k) \tag{2}$$

In the first scenario we gauge the effectiveness of the proposed traffic-based bandwidth allocation algorithm. We consider, in line with [1,2], the scenario illustrated in Figure 2, that consists of a single-bottleneck with 2 core nodes, 6 access nodes, 40 end nodes (20 source-destination pairs) and traffic conditioners at the edge. Each ingress conditioner is configured with one profile for each traffic source, and drops out-of-profile packets. All links are full-duplex and have a propagation delay of 1 ms. The capacity of the link connecting the two core nodes is equal to 6 Mb/s, that of the links connecting the access nodes to core nodes is equal to 10 Mb/s, and that of the links connecting the end nodes to access nodes is 2 Mb/s. The buffer size of each link can contain 50 packets.

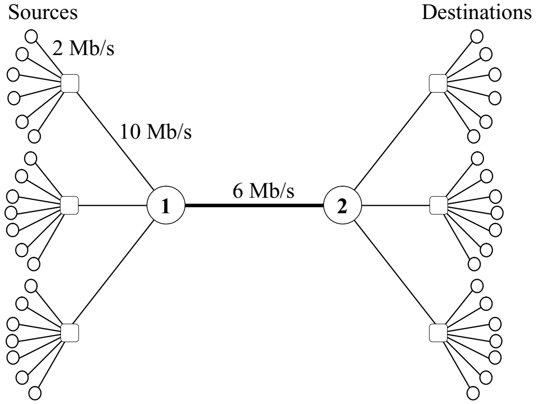


Fig. 2. Network topology with a single bottleneck

We use 20 Exponential On-Off traffic sources; the average On time is set to 200 s, and the average Off time is varied in the 0 to 150 s range to simulate different traffic load conditions while varying at the same time the percentage of bandwidth left unused by every connection. During On times each source transmits with a constant rate that we refer to hereafter as the source's peak rate.

Six sources have a peak rate of 50 kb/s and a subscribed rate of 150 kb/s, 8 sources have a peak rate of 250 kb/s and a subscribed rate of 200 kb/s, while the remaining six sources have a peak rate of 1 Mb/s and a subscribed rate of 500 kb/s; the minimum bandwidth required by each source, r_min_k, is equal to 10 kb/s. The algorithm updating interval, T_u, is set to 20 s and γ is set to 0.9.

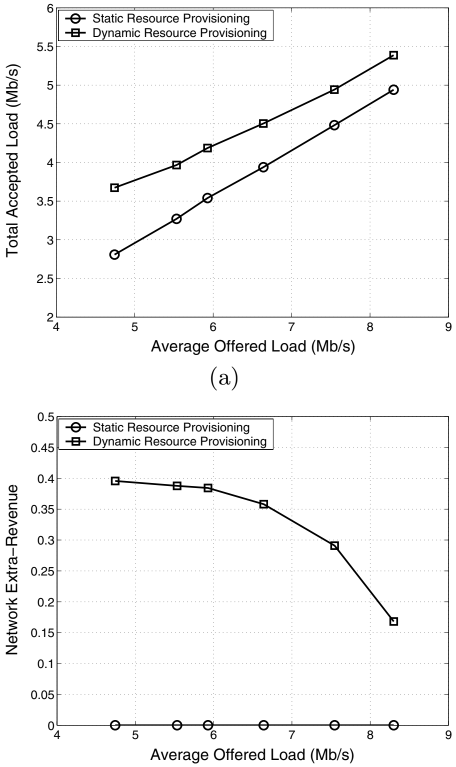
We assume, for simplicity, that all users have the same weight w_k and the same utility function proposed in [9], $U_k(x) = 0.5 \cdot log(1 + x)$, that models the perceived utility of elastic traffic for an allocation of x bandwidth units.

Note that a realistic characterization of network applications is outside the scope of this paper. The specification of the utility function allows us exclusively to gauge the extra network revenue that can derive from the deployment of our proposed bandwidth allocation algorithm.

Figures 3(a) and 3(b) show, respectively, the average total load accepted in the network and the corresponding total extra-revenue as a function of the average total load offered to the network.

It can be observed that our dynamic provisioning algorithm is very efficient in resource allocation compared to a static provisioning algorithm for all values of the offered load, providing improvements up to 31% in the total accepted traffic.

The maximum network extra-revenue is achieved when the average Off time of Exponential sources is equal to 150 s, corresponding to an offered load approximately equal to 5 Mb/s. In this situation, in fact, the average number of idle connections (i.e. 9) is sufficiently high to exalt our dynamic allocation algorithm that re-allocates unused bandwidth to active users who can take advantage of it, sending extra-traffic and generating network extra-revenue. With lower Off time values (i.e. with higher offered loads) the total revenue slightly decreases as less



(a)

(b)

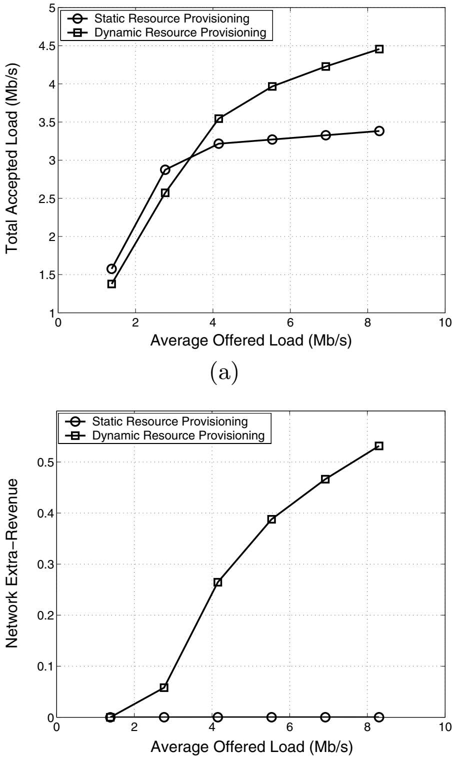
Fig. 3. Average total accepted load (a) and network extra-revenue (b) versus the average load offered to the network of Figure 2

connections are idle, in average, and consequently less bandwidth is available for re-allocation.

To investigate the impact on the performance of the update interval duration, we have considered, in the same scenario, different values for T_u, viz. 40 s and 60 s. We found that the average increase in the total accepted load, expressed as a percentage of the traffic admitted in the static allocation case, is of 9% for $T_u = 40$ s and 7% for $T_u = 60$ s, while for $T_u = 20$ s it was 16% (see Figure 3(a)). These results allow to gauge the trade-off between performance improvement and overhead resulting from a more frequent execution of the allocation algorithm.

In the same scenario of Figure 2 we then fixed the average Off time of Exponential sources to 100 s while maintaining the average On time equal to 200 s, and we varied the peak rate of all sources scaling them by a factor α, with $0.25 \leq \alpha \leq 1.5$. Figures 4(a) and 4(b) show the total accepted load and the total extra-revenue in this scenario.

At very low load the static provisioning technique achieves slightly higher performance than dynamic provisioning. This is due to the fact that in this situation static provisioning is in effect sufficient to accommodate all incoming traffic; on the other hand, dynamic provisioning needs some time (in the worst case up to T_u seconds) to track the transition of sources from the idle to the active state. For all other traffic loads the advantage of the proposed dynamic bandwidth allocation algorithm is evident both in terms of accepted load and network extra-revenue.



(a)

(b)

Fig. 4. Average total accepted load (a) and network extra-revenue (b) versus the average load offered to the network of Figure 2

We then considered the network topology proposed in [9] and illustrated in Figure 5. This network scenario is more complex than the previous one and it allows to test our proposed allocation algorithm in a more realistic core network topology. It comprises 11 core nodes, 8 access nodes, 36 end nodes (18 source-destination pairs) and 28 bidirectional links, all having the same propagation delay, equal to 5 ms. The capacities are given next to the links in the Figure. Eighteen Exponential On-Off connections share the network. Table 2 reports the peak rate, the subscribed rate and the path for all the connections, which are the same as in [9].

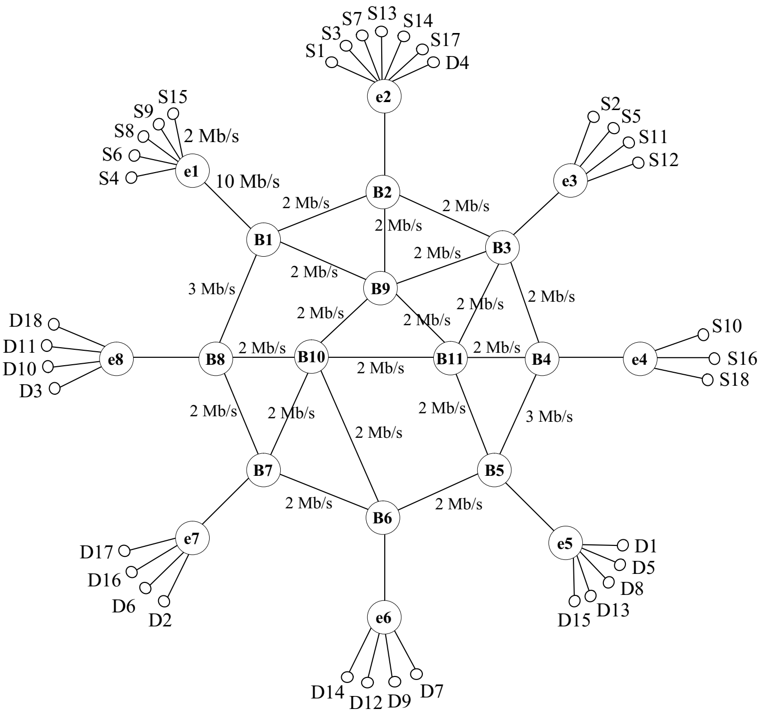
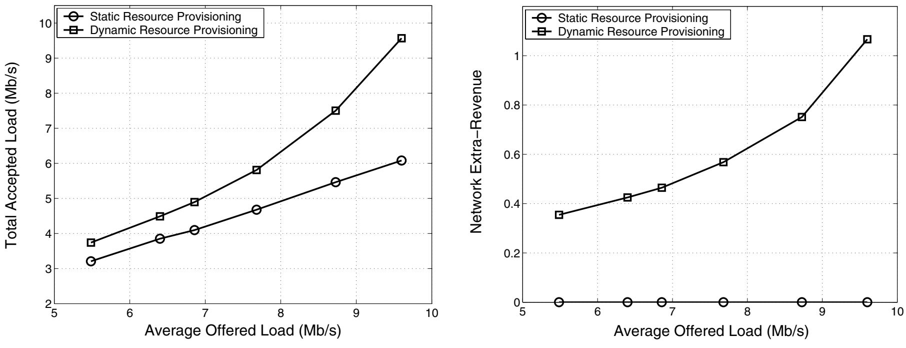


Fig. 5. Complex core network topology



(a) (b)

Fig. 6. Average total accepted load (a) and network extra-revenue (b) versus the average load offered to the complex core network of Figure 5

Figures 6(a) and 6(b) show the performance of the considered bandwidth allocation algorithm as a function of the total load offered to the network. The results are in line with those achieved with the single-bottleneck scenario and show that our proposed allocation algorithm allows to increase both total accepted traffic and network revenue with respect to a static allocation technique.

Table 2. Peak rate, subscribed rate and path for the connections in the network scenario of Figure 5

Connection	Peak Rate (kb/s)	Subscribed Rate (kb/s)	Path
1	100	300	e2-B2-B3-B4-B5-e5
2	100	300	e3-B3-B9-B10-B7-e7
3	100	300	e2-B2-B1-B8-e8
4	100	300	e1-B1-B2-e2
5	100	300	e3-B3-B4-B5-e5
6	100	300	e1-B1-B8-B7-e7
7	500	400	e2-B2-B9-B10-B6-e6
8	500	400	e1-B1-B9-B11-B5-e5
9	500	400	e1-B1-B8-B7-B6-e6
10	500	400	e4-B4-B11-B10-B8-e8
11	500	400	e3-B3-B2-B1-B8-e8
12	500	400	e3-B3-B4-B5-B6-e6
13	1000	500	e2-B2-B3-B4-B5-e5
14	1000	500	e2-B2-B9-B10-B6-e6
15	1000	500	e1-B1-B9-B11-B5-e5
16	1000	500	e4-B4-B5-B6-B7-e7
17	1000	500	e2-B2-B1-B8-B7-e7
18	1000	500	e4-B4-B11-B10-B8-e8

6 Conclusion

In this paper we proposed a novel service model where users subscribe for guaranteed transmission rates, and the network periodically individuates unused bandwidth that is re-allocated and guaranteed with short-term contracts to users who can better exploit it. We described a distributed dynamic resource provisioning architecture for quality of service networks. We developed an efficient bandwidth allocation algorithm that takes explicitly into account traffic statistics to increase the users perceived utility and the network extra-revenue.

Simulations results measured in realistic network scenarios show that our allocation algorithm allows to increase both resource utilization and network revenue with respect to static provisioning techniques.

References

1. A. T. Campbell and R. R.-F. Liao. Dynamic Core Provisioning for Quantitative Differentiated Services. *IEEE/ACM Transactions on Networking*, pages 429–442, vol. 12, no. 3, June 2004.
2. H. Schulzrinne and X. Wang. Incentive-Compatible Adaptation of Internet Real-Time Multimedia. *IEEE Journal on Selected Areas in Communications*, pages 417–436, vol. 23, no. 2, February 2005.

3. T. Ahmed, R. Boutaba, and A. Mehaoua. A Measurement-Based Approach for Dynamic QoS Adaptation in DiffServ Network. *Journal of Computer Communications, Special issue on End-to-End Quality of Service Differentiation, Elsevier Science*, 2004.
4. M. Mahajan, M. Parashar, and A. Ramanathan. Active Resource Management for the Differentiated Services Environment. *International Journal of Network Management*, pages 149–165, vol. 14, no. 3, May 2004.
5. F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, pages 33–37, vol. 8, 1997.
6. J. Aweya, M. Ouellette, and D. Y. Montuno. Design and stability analysis of a rate control algorithm using the Routh-Hurwitz stability criterion. *IEEE/ACM Transactions on Networking*, pages 719–732, vol. 12, no. 4, August 2004.
7. D. Bertsekas and R. Gallager. *Data Networks, 2nd Edition*. Prentice-Hall, 1992.
8. L. Breslau and S. Shenker. Best-Effort versus Reservations: A Simple Comparative Analysis. In *Proc. ACM SIGCOMM*, pages 3–16, September 1998.
9. R. J. La and V. Anantharam. Utility-Based Rate Control in the Internet for Elastic Traffic. *IEEE/ACM Transactions on Networking*, pages 272–286, vol. 10, no. 2, April 2002.

Artificial Intelligence Techniques in the Dynamic Negotiation of QoS: A User Interface for the Internet New Generation

Zeina Jrad[1], Francine Krief [2], Lahcene Dehni[1], and Younès Bennani[1]

[1] LIPN Laboratory, University of Paris13
99 Avenue J-B Clément, 93430 Villetaneuse, France
{zj, ldehni, younes.bennani}@lipn.univ-paris13.fr
[2] LaBRI Laboratory, UMR CNRS 5800
University of Bordeaux I, cours de la Libération, 33400 Talence, France
francine.krief@labri.fr

Abstract. The Internet New Generation provides a service adapted to the needs of the applications (particularly real time) including a Quality of Service (QoS) guaranteed. It is based on the DiffServ architecture and the policy-based networking management. It also uses the agent technology to solve problems and control infrastructures and flows. Access to the Internet New Generation is more difficult. The user has to choose the best service provider and indicate the technical parameters that he needs.

In this paper, we investigate the use of some techniques of the AI (Artificial Intelligence) domain to implement a user interface in order to help the user access to the Internet New Generation. We use the connectionist clustering in the management of the negotiation profiles. Then we use the agent technology to help the user to choose the best service provider, dynamically negotiate the SLS on the user's behalf, follow the users behavior to be able to anticipate the negotiations and manage the re-negotiations.

1 Introduction

In recent years, the development and the apparition of real time applications as well as multimedia applications have witnessed an exponential increase. The real time constraints of these applications present a big challenge for their integration. That's why we need services adapted to specific application needs with a guaranteed Quality of Service (QoS) [1]. The Internet New Generation has to provide these services particularly for real time applications.

To provide QoS in best-effort IP networks, the IETF developed the IntServ and DiffServ architectures. The IntServ model was created to transport audio, video, real time data as well as traditional data traffic in a way similar to the one of an integrated services network [2]. The DiffServ model was created because of the difficulties of deployment of IntServ [3],[4]. The Internet New Generation uses the DiffServ architecture to provide QoS.

However, the implementation of QoS mechanisms is a very heavy task. It is difficult to manually configure all the network devices because of the abundance of

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 146 – 158, 2006.

QoS information and because of the dynamic nature of QoS configurations. The operator must control the attribution of network resources according to applications and users characteristics. Using management tools adapted to QoS quickly proves essential. In order to simplify the router's configuration by permitting its automation, the IETF proposed a general framework called policy-based networking [5] for the control and management of these IP networks. The Internet New Generation is based on the policy-based networking management.

Most applications cannot dynamically express their QoS requirements to obtain the adapted level of service. For each application, the customer and the provider have to agree on rules of assignment of service levels. They sign a contract called SLA (Service Level Agreement) which is then translated into high-level policies. These policies are not directly executable by the network devices. They must be translated into intermediate and then into low level policies which are understandable by network devices. The SLS (Service Level Specification) is the technical version of the SLA [6].

There are some protocols that allow the dynamic negotiation [7], [8] and [9] of the required level of service and the needed quality between the user and the network entities. However, this negotiation seems to be complex because the user has to indicate himself the technical parameters that reflect the required quality of service. The difficulty of the process can be reduced by replacing the user in finding applications needs in terms of QoS parameters according to the context of use.

In this context, the user needs to be assisted, first in choosing the best provider, and then in dynamically negotiating the technical parameters of the SLA (the SLS).

In this paper, we propose an intelligent user interface in order to help the user access to the Internet New Generation. We investigate the use of some techniques of the AI (Artificial Intelligence) domain to implement this interface. We use the connectionist clustering in the management of the user's profiles. Then we use the agent technology to help the user in the negotiation of SLS parameters according to the current profile.

The paper is organized as follows. Section 2 describes the framework of the proposed interface. The third section describes the first layer of the proposed model (the profile management layer). Sections 4 and 5 describe the control and negotiation layer respectively. Finally we present future work in section 6.

2 The Proposed Framework

The user assistant proposed in this work is placed on the user terminal and is called NIA (Negotiation Individual Assistant). The NIA negotiates the quality of service between the user and the service provider, from one side, and between the user and the network, from the other side (Fig.1). The main purpose of the assistant is the representation of the user in requesting and negotiating the desired quality of service in a dynamic environment. This representation is illustrated in the following points:

- The analysis of the user's work for a profile attribution.
- The save and update of all data concerning the user preferences.
- The translation of the user's requests in SLS parameters.
- The negotiation of the desired QoS with a service provider.
- The monitoring of the real-time quality to compare it with the negotiated one.
- The substitution of the user in the decisions-making.

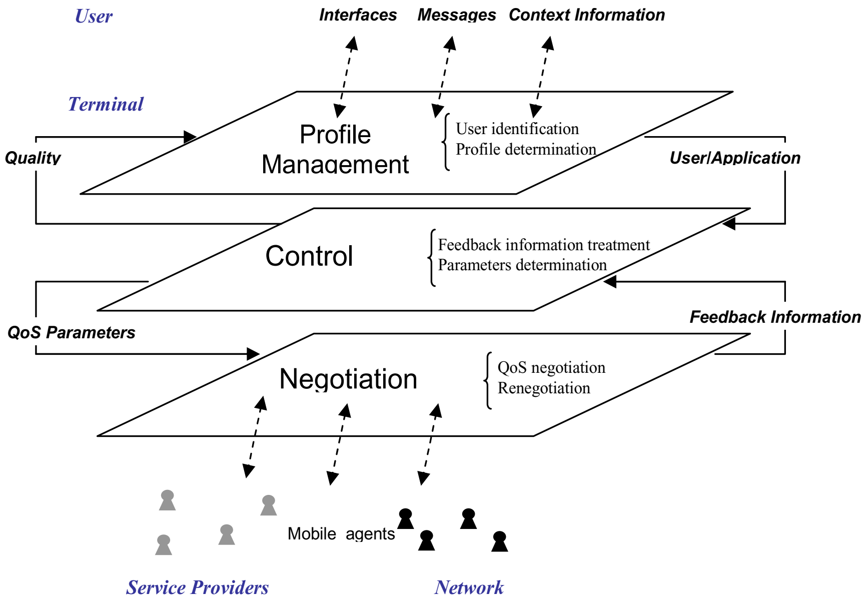


Fig. 1. The layers of the proposed framework

As shown in Fig. 1, the proposed assistant contains different layers. The first one is the profile management layer. This is the layer of direct contact with the user. Once connected, the principal task of the modules of this layer is to react autonomously in order to follow the user's work. User and terminal contexts are saved along with the used applications and their requirements in the knowledge base of the system. These data are modified systematically according to any change in the user choices and actions. The reasoning modules will also use them in order to deduce a general profile that represents the user.

The second layer is for the control. Once the user's needs and preferences are identified, the next step consists of verifying that these preferences are converted into the appropriate SLS values.

The control made in this layer should guarantee a good adaptation of the attributed quality with the real-time user's work independently of the user's mobility or even the user's profile variation. The reasoning mechanism needed at this level is a permanent comparison of input data from the profile manager layer (profiles, characteristics) and those from negotiation layer (degradation or modification of quality, cancellation of contract between the two sides, new propositions of services). The output of this layer will be SLS parameters values sent to the negotiation layers. These values represent the users and applications requirements.

The third layer is the Negotiation layer. This layer manages the service publication, subscription, selection and negotiation.

3 Profile Management Layer

This layer is introduced to identify the user and to analyze his work. User preferences and application requirements are saved in the knowledge base of the system. These data are modified systematically according to any change in the user choices and actions. Applications can be classified in many categories according to their needs (delay, jitter…) and to the type of the supported information (data, voice, image…). The profile of the application will then be determined according to these categories and to the requirements of the user.

As an input to this layer, we can identify two types of data:
Information collected through a communication with the user via graphical interfaces or messages.

Information collected through an observation of the user's behaviour.
Once the information is analyzed, the result is a user profile that represents the user's preferences in terms of quality and an application profile that describes the needs of each application.

So this layer plays the role of intermediary between the user and the system. It defines the graphical interfaces needed for the communication with the user. It may sends him messages or questions and help him choosing the answers that best match with his profile or needs.

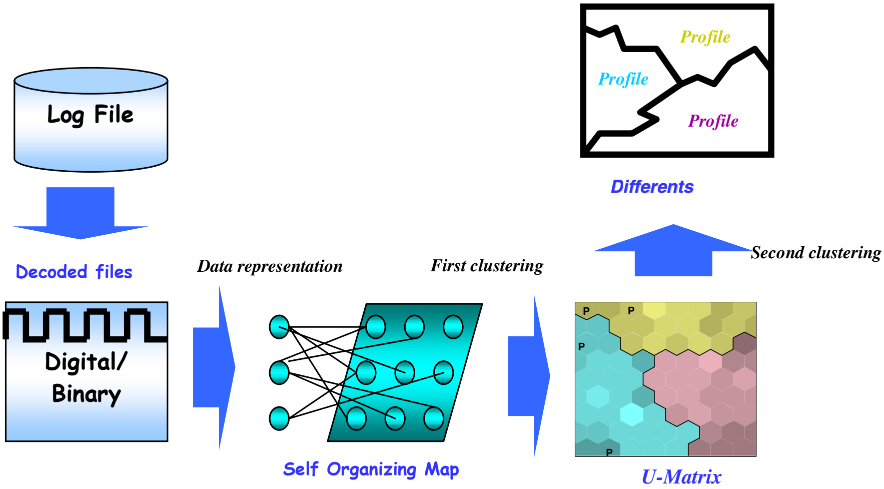


Fig. 2. Profile management procedure

Our approach, represented in Fig. 2, consists in recovering, first of all, data that represent traces of use (i.e. log files [10]). These data will be cleaned and re-coded in a numerical or binary format to be easily treated. We build then a Self Organizing Map from the re-coded file in order to extract profiles [10]. Finally, we carry out a classification to better see the clusters structure of the map, followed by a segmentation of the SOM in order to separate the different profiles.

The steps of our approach represented in Fig. 2 are detailed in the following subsections.

3.1 The Unsupervised Connectionist Learning

The unsupervised numerical learning, or automatic classification, consists in determining a partition of an instances space from a given set of observations, called training set. It aims to identify potential trend of data to be gathered into classes. This kind of learning approach, called clustering, seeks for regularities from a sample set without being driven by the use of the discovered knowledge. Euclidian distance is usually used by clustering algorithms to measure similarities between observations.

Self-Organizing Maps (SOM) implement a particular form of competitive artificial neural networks [11]; when an observation is recognized, activation of an output cell – competition layer – leads to inhibit activation of other neurons and reinforces itself. It is said that it follows the so called "Winner Takes All" rule. Actually, neurons are specialized in the recognition of one kind of observations. The learning is unsupervised because neither the classes nor their numbers are fixed a priori.

This type of neural networks is organized in a two dimensional layer of neurons [12]. Each neuron *k* is connected to *n* inputs through *n* exciter connections of respective weights w and to their neighbors with inhibiting links.

The training set is used to organize these maps under topological constraints of the input space. Thus, a mapping between the input space and the network space is constructed; closed observations in the input space would activate closed units of the SOM.

An optimal spatial organization is determined by information received from the neural networks. When the dimension of the input space is lower than three, both of the position of weights vectors and the direct neighborhood relations between cells can be visually represented. Thus, a visual inspection of the map provides qualitative information on its architecture.

The connectionist learning is often presented as a minimization of a risk function [13]. In our case, it will be carried out by the minimization of the distance between the input samples and the map prototypes (referents), weighted by a neighborhood function h_{ij}. To do that, we use a gradient algorithm. The criterion to be minimized is defined by:

$$E_{SOM} = \frac{1}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} h_{jNN(x^{(k)})}\left\|w_{.j} - x^{(k)}\right\|^{2} \tag{1}$$

Where N represents the number of learning samples, M the number of neurons in the map, $NN(x^k)$ is the neuron having the closest referent to the input form x^k, and h the neighborhood function. The neighborhood function h can be defined as:

$$h_{rs} = \frac{1}{\lambda(t)}\exp\left(-\frac{d_1^2(r,s)}{\lambda^2(t)}\right) \tag{2}$$

(t) is the temperature function modelling the neighborhood extent, defined as:

$$\lambda(t) = \lambda_i \left(\frac{\lambda_f}{\lambda_i} \right)^{\frac{t}{t_{\max}}} \tag{3}$$

$_i$ and $_f$ are respectively initial and the final temperature (for example $_i$= 2, $_f$ = 0.5). t_{max} is the maximum number allotted to the time (number of iterations for the *x* learning sample). $d_1(r,s)$ is the Manhattan distance defined between two neurons r and s on the map grid, with the coordinates *(k,m)* and *(i,j)* respectively:

$$d_1(r,s) = |i-k| + |j-m| \tag{4}$$

The learning algorithm of this model proceeds essentially in three phases:

- Initialization phase where random values are assigned to the connections weights (referents or prototypes) of each neuron of the map grid.
- Competition phase during which, for any input form $x^{(k)}$, a neuron $NN(x^k)$, with neighborhood, $V_{NN(x^{(k)})}$ is selected like a winner. This neuron has the nearest weight vector by using Euclidean distance:

$$NN(x^{(k)}) = \underset{1 \le i \le M}{\operatorname{argmin}} \left\| w_{.i} - x^{(k)} \right\|^2 \tag{5}$$

- Adaptation phase where the weights of all the neurons are updated according to the following adaptation rules:

If $w_{.j} \in V_{NN(x^{(k)})}$ then adjust the weights using:

$$w_{.j}(\mathrm{t}+1) = w_{.j}(\mathrm{t}) \text{-} \varepsilon(t) h_{jNN(x^{(k)})} \left(w_{.j}(\mathrm{t}) \text{-} x^{(k)} \right) \tag{6}$$

else

$$w_{.j}(\mathrm{t}+1) = w_{.j}(\mathrm{t}) \tag{7}$$

Repeat this adjustment until the SOM stabilization.

3.2 SOM Map Segmentation

We segment the SOM using the K-means method (Fig. 3). It is another clustering method that consists in arbitrarily choosing a partition; the samples are then treated one by one. If one of them becomes closer to the center of another class, it is moved into this new class. We calculate the centers of new classes and we reallocate the samples to the partitions. We repeat this procedure until having a stable partition.
The criterion to be minimized in this case is defined by:

$$E_{K-means} = \frac{1}{C}\sum_{k=1}^{C}\sum_{x\in Q_k}\left\|x - c_k\right\|^2 \tag{8}$$

Where C represents the number of clusters, Q_k is the cluster k, C_k is the center of the cluster Q_k or the referent.

The basic algorithm requires fixing K, the number of wished clusters. However, there is an algorithm to calculate the best value for K assuring an optimal clustering. It is based principally on the minimization of Davies-Bouldin index [14], defined as follows:

$$I_{DB} = \frac{1}{C}\sum_{k=1}^{C}\max_{l\neq k}\left\{\frac{S_c(Q_k)+S_c(Q_l)}{d_{ce}(Q_k,Q_l)}\right\} \tag{9}$$

With,

$$S_c(Q_k) = \frac{\sum_i \left\|x_i - c_k\right\|^2}{\left|Q_k\right|} \tag{10}$$

$$d_{cl}(Q_k,Q_l) = \left\|c_k - c_l\right\|^2 \tag{11}$$

C is the number of clusters, S_c is the intra-cluster dispersion, and d_{cl} is the distance (centroid linkage) between the clusters centers k and l. This clustering procedure aims to find internally compact spherical clusters which are widely separated.

There are several methods to segment the SOMs [15]. Usually, they are based on the visual observations and the manual assignment of the map cells to the clusters. Several methods use the K-means algorithm with given ranges for K value. Our work is based on the approach of Davies-Bouldin index minimization.

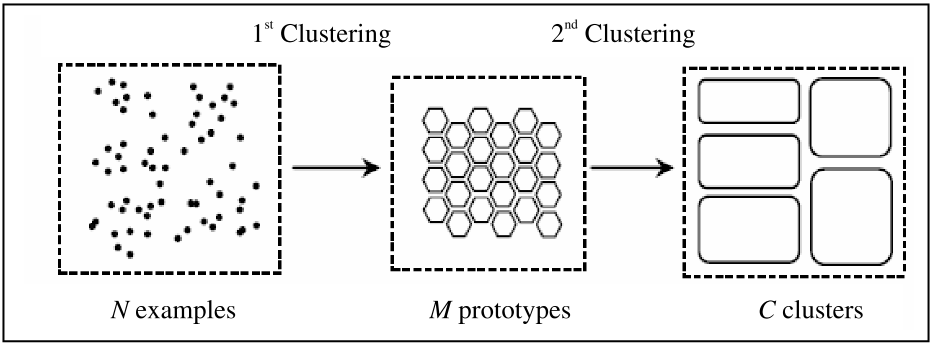


Fig. 3. Two successive clusterings: SOM followed by K-means

We note that the K-means approach can be directly applied to the data instead of SOMs approach. In our work, we applied it to the SOMs results. The idea is to use SOMs as a preliminary phase in order to set a sort of data pre-treatment (dimension reduction, regrouping, visualization...). This pre-treatment has the advantage to reduce the clusters calculation complexity and also ensures a better visualization of the automatic classification results.

Moreover, the use of SOMs for visualization is crucial, especially in the case of data multivariate: dimension > 2 or 3. In this last case, the SOMs permit, on one hand, to reduce the data space dimension, and on the other hand, to visualize the clusters in the plan.

3.3 Simulations Results

We applied the two algorithms described above on our data (log files describing different traces of use) in order to determine the negotiation profiles. In the simulations, we used the SomToolbox proposed by the researchers of the HUT (Helsinki University of Technology) of the T. Kohonen team [16]. The results obtained are very promising (Fig. 4).

Fig. 4.a. is a representation of a SOM map seen as “Component Planes” that allows the visualization of the partition of the different variable values. The highest values of the variables are in red and the lowest values are in blue. This representation allows us to identify the clusters structure of the map.

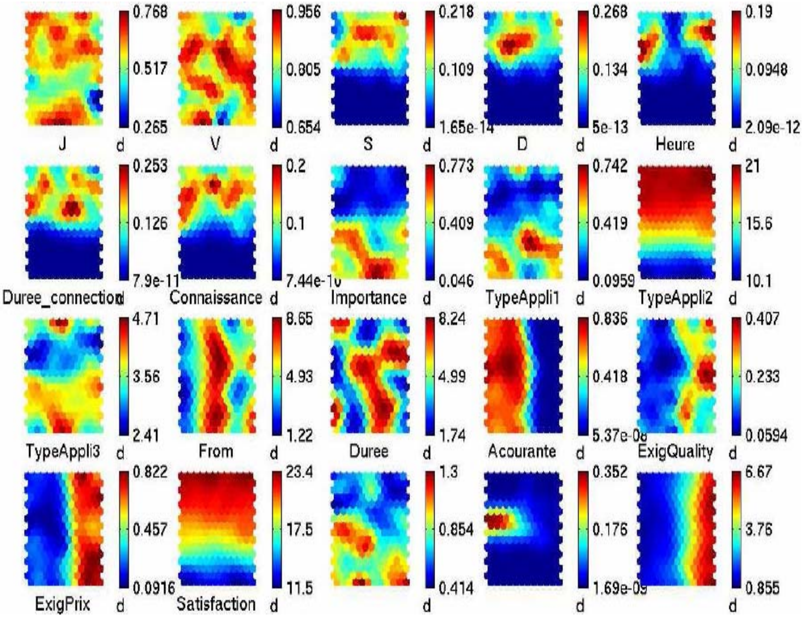


Fig. 4. a. SOM clustering

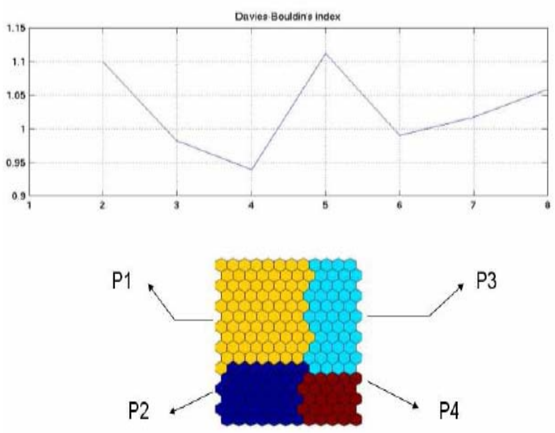


Fig. 4.b. Resulting profiles

Fig. 4.b. represents the neurons segmentation of SOM map (second clustering). The curve shows that the minimal value of the index of Davies-Boudin corresponds to an optimal clustering resulting in four profiles. The various colors represent the different clusters or the identified negotiation profiles.

The classes finally obtained are coherent although they result from an unsupervised classification without any pre-established class before the treatment. These results are thus encouraging because they make it possible to interpret the obtained profiles.

4 Control Layer

Once the user's needs and preferences are identified in a negotiation profile, the next step consists in verifying that these preferences are converted into the appropriate parameters values [17].

The control made in this layer should also guarantee a good adaptation of the attributed quality with the real-time user's work independently of the user's mobility or even the user's profile variation. The reasoning mechanism needed at this level is a permanent comparison of input data from the profile manager layer (profiles, characteristics) and those from the negotiation layer (degradation or modification of quality, cancellation of contract between the two sides, new propositions of services). The output of this layer will be QoS parameters values sent to the negotiation layer. These values are related to the negotiation profile determined in the previous layer and represent the users and applications requirements. The control layer accomplishes these functionalities:

It determines the values that should be attributed to all of the negotiation parameters depending on the constraints of the system, the service providers and the profile of negotiation.

It establishes the link between the user and the service provider. It verifies that both of the two parts respect the negotiated services. The satisfaction of the user is deduced from the analysis of his behavior.

It analyses the feedback information collected by the responsive agents in order to take decisions concerning the re-negotiation and to evaluate the state of the link and the video packet transmission.

5 Negotiation Layer

The Negotiation layer is responsible for the negotiation of the QoS parameters received from the control layer. It sends the QoS parameters to the network entities and starts the process of negotiation. It may also ask for a change in the required services according to the needs of the user and the applications. This corresponds to a request of re-negotiation between the two parts. Mobile agents, [18] and [19], are sent to the service providers in order to bring new offers. Three types of agents are proposed [20]:

1) User Negotiator (UN): This is a mobile agent that is sent by the User Overseer (UO) on the platform of the provider (Access Mediator). Its first task is to survey the offers made by the providers. This is the discovery phase. The second phase is the negotiation which takes place when the user has a specific need.

2) Access Negotiator (AN): It is created by the provider to each mobile agent (UN) that arrives on its platform in order to negotiate services in favor of the provider.

3) User Overseer (UO): It manages the entire negotiation process on behalf of the user. It sends UN for the service survey and negotiation. It collects the results of the different negotiation threads and then makes the final decision.

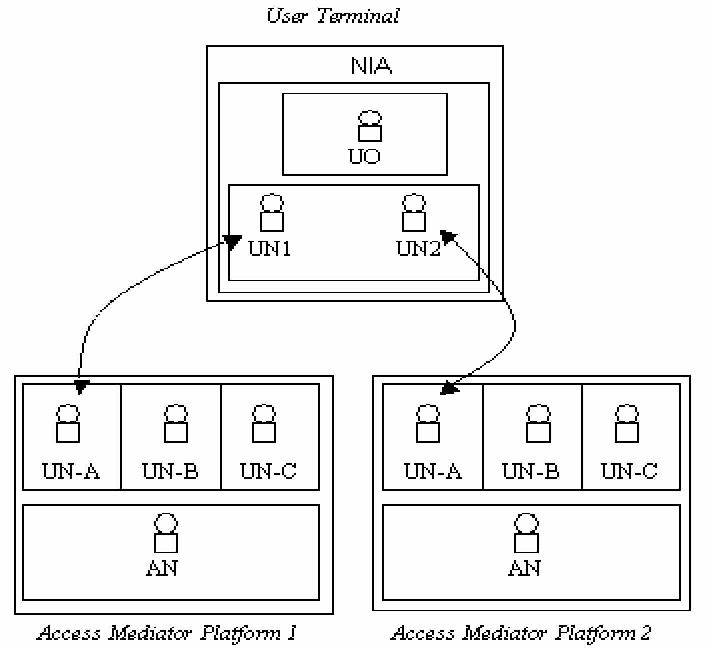


Fig. 5. Framework for the dynamic negotiation of SLA/SLS

A possible solution would be to create a marketplace where all providers could propose their offers and negotiate with interested customers. However this solution asks a high degree of cooperation between providers. From the Telecommunications point of view, this is not the preferred solution. We based our solution on the assumption that providers are mostly competitors.

5.1 The Negotiation Process

The proposed solution has two phases. A survey phase, when the user agent discovers the new offers made by the providers and a negotiation phase which takes place when the client has specific QoS needs.

In the first phase, every provider opens access to a multi-agent platform on its site that would welcome customer's UN agents. The User Overseer (UO) sends a User Negotiator (UN) to the provider's platform with which he has a predefined contract. The UN takes the user profile which corresponds to the QoS characteristics of the applications the user often use. The access to the platform requires an authentification phase. UN then starts to survey the offers made by the provider. When new offers are proposed, UN filters them and sends them to his UO, the one that matches the user's profile. The mobile agents stay on the provider's platforms and continuously survey the publication of new offers.

The second phase starts when a customer has a specific need. The UO filters the offers sent by the UN and sends its needs to the concerned UNs. Then the UNs open negotiations with their AN counterparts to obtain the best rates and quality of services. When the UN and the AN reach an agreement (or after a certain period of time where no agreement is done), the UN sends to the Overseer the description of the best proposal made by the AN. Then the Overseer compares the different offers it received from the different ANs and sends back an agreement to the best one. Then every agent concerned by the different concurrent negotiations returns to its normal state.

To implement our model and protocol of negotiation, we used the platform JADE (Java Agent Development Framework). JADE [21] is a software Framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI.

More details concerning the implementation of our model and the protocol of negotiation can be found in [22] and [23].

6 Conclusion and Future Work

The Negotiation Individual Assistant (NIA) presented in this paper constitutes an interface between the user and the network in the context of the Internet of new generation. This interface integrates two interesting techniques of the Intelligence Artificial domain: The connectionist learning and the agent technology.

A clustering algorithm, based on the topological Self Organizing Maps (SOMs), is used (in the first layer of the NIA) to determine a negotiation profile that represents the user preferences and the applications needs. This profile is than used, in the second layer of the NIA, in order to find the appropriate values for SLS parameters.

On the other side, the agent technology is used (in the third layer of the NIA) in order to help the user to choose the best service provider, dynamically negotiate the SLS on the user's behalf, follow the user's behavior to be able to anticipate the negotiations and manage the re-negotiations. The Introduction of a multi-agents system, in both of the user and providers sides, has shown a good performance in the choice of the best service provider. The agents of our system communicate via a FIPA protocol [24]. This approach has many advantages. The terminal charge is reduced, the system can function on a large range of terminals and the service providers can more easily propose new services.

Our NIA has been implemented and tested in two French national research projects: ARCADE [25] [26] and IPSIG [27].

Future work may concern different topics:

- The introduction of security levels beyond the selection criteria of the best service;
- The introduction of the mobility of the user;
- The application of the future mobile terminals. Some terminals will also be able to use different access technologies either simultaneously or one at time. Therefore the most important property of any communication system is its ability to handle mixtures of flows and traffic characteristics in a reasonable way [28];

References

1. QoSforum, "QoS protocols & architectures", White Paper, July 1999.
2. Braden R., Clark D. and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.
3. Blake S., D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.
4. Nichols K., S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
5. Yavatkar R., D. Pendarakis, R. Guerin, "A Framework for Policy Based Admission Control", RFC 2753, January 2000.
6. D. Goderis and al., Service Level Specification Semantics and Parameters, draft-tequila-sls-02.txt, Internet draft, January, 2002.
7. J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Raja, A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000.
8. K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, A. Smith, " COPS Usage for Policy Provisioning", March 2001.
9. T.M.T. Nguyen, , N. Boukhatem, Y. El Mghazli, N. Charton, Louis-Louis Hamer, G. Pujolle, "COPS-PR usage for SLS negotiation (COPS-SLS)", draft nguyen-rap-cops-sls-03.txt, Internet Draft, July 2002.
10. K. Benabdeslem, Approches Connexionnistes pour la visualisation et la classification des données issues d'usages de l'Internet, , thesis in computer science, LIPN, University of Paris13, France, December, 2003.
11. T. Kohonen and S. Kaski and H. Lappalainen, Self-Organized Formation of Various Invariant-Feature Filters in the Adaptive-Subspace SOM, pages 1321-1344, Neural Computation, 1997.

12. F. Zeharoui and Y. Bennani, M-SOM-ART : Growing Self Organizing Map for sequence clustering and classification, European Conference on Artificial Intelligence (ECAI), Valencia, Spain, August, 2004
13. Y. Bennani, Réseaux de neurones artificiels, Chapter in « Encyclopedie d'Informatique et Science de l'information », Edition Vuibert, 2005.
14. Davies, D., L., Bouldin, D., W., : A Cluster Separation Measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1(2): pp. 224-227. (1979).
15. Juha, A. , Esa, A. : Clustering of the Self-Organizing Map. IEEE Tractions On Neural Networks volume 11, n° 3, (2000).
16. E. Alhoniemi and J. Himberg and J. Parhankangas and J. Vesanto, SOM Toolbox, 2000, Copyright (C), http://www.cis.hut.fi/projects/somtoolbox/.
17. Z. Jrad. And F. Krief. An Intelligent Interface for the Dynamic Negotiation of QoS in ARCADE. ACS/IEEE International Conference on Pervasive Services. ICPS'2004. Beirut, Liban. Juillet 2004.
18. J.-P. Briot, Y. Demazeau, "principles and architecture of MAS", 2002.
19. N. Agoulmine, M. Fonseca, A. Marshall, "Multi-domain Policy Based Management Using Mobile Agents, Lecture Notes in Computer Science, 2000.
20. G. Klein and F. Krief. Mobile Agents for Dynamic SLA Negotiation. International Workshop on Mobile Agents for Telecommunication Applications. MATA'2003. Lecture Notes on Computer Science, Springer. Marrakech, Maroc. October 2003.
21. JADE-A FIPA-compliant agent framework, F. Bellifemine and A. Poggi and G. Rimassa, pages 97-108, Proceedings of the Practical Applications of Intelligent Agents and Multi-Agents, April, 1999.
22. Zeina JRAD, Thesis report, Apports des techniques de l'Intelligence Artificielle dans la négociation dynamique de la qualité de service : Proposition d'un assistant à l'utilisateur dans les réseaux IP de nouvelle génération, LIPN, University of Paris 13, France, May 2006.
23. Z. Jrad., B. Benmammar, J. Correa, F. Krief , N. Mbarek. A user assistant for QoS negotiation in a dynamic environment using agent technology. Second IFIP International Conference on Wireless and Optical Communications Networks WOCN 2005. Dubai, United Arab Emirates UAE, March 2005.
24. http://www.fipa.org, 2005
25. K. AlAgha and W. Dabbous and Y. Ghamri Doudanea and Z. Jrad and F. Krief and F. Le Garrec and S. Masson and P. Minet and T.M.T. Nguyen and G. Pujolle and R. Rizo and R. Serban, Final report of Projet ARCADE, Janvier,2003.
26. Projet Arcade, Architecture de Control Adaptative des Environments IP, Web Site: http://www-rp.lip6.fr/arcade/, 2002.
27. Projet IpSig, Signalisation générique du monde IP, http://www.telecom.gouv.fr/rnrt/rnrt/projets/res{_}02{_}85.htm, 2004.
28. Middleware for Mobile Applications Beyond 3G, Kimmo Raatikainen, SmartNet 2002.

An Approach to Integrated Semantic Service Discovery

Shanshan Jiang and Finn Arve Aagesen

Department of Telematics
Norwegian University of Science and Technology (NTNU)
N-7491 Trondheim, Norway
{ssjiang, finnarve}@item.ntnu.no

Abstract. In a distributed service environment, service discovery is a core functionality to locate the desired services. We propose an integrated semantic service discovery approach based on ontology, which provides matching of functional and non-functional properties. Functional properties are described in terms of operations, inputs, outputs, preconditions and effects, while non-functional properties are specified as business policies, QoS properties and context policies. Ontological inference and rule-based reasoning are applied for automatic and accurate discovery.

1 Introduction

In a distributed service environment, service discovery is a core functionality to locate the desired services. *Service discovery* is a process of finding the desired service(s) by matching *service descriptions* against *service requests*. A *service description* provides service-related information which can be advertised by a service provider and searched during service discovery process. Such information usually includes functional properties and non-functional properties. In this paper, functional properties representing functionality of a service are modeled in terms of operations, inputs, outputs, preconditions and effects, while non-functional properties comprise business policies, Quality of Service (QoS) properties as well as context policies. QoS properties include QoS parameters and QoS policies. A *service request* represents user's service requirements, comprising requirements on functional and non-functional properties.

Ontologies are the basis for adding semantic expressiveness to service descriptions and requirements. An *ontology* is an explicit and formal specification of a shared conceptualization [19]. A service ontology is accordingly an explicit and formal specification of core concepts of the functional and non-functional properties of service. "A *domain ontology* (or *domain-specific ontology*) models a specific domain and represents the particular meanings of terms as they apply to that domain. An *upper ontology* is a model of the common objects that are generally applicable across a wide range of domain ontologies." [1] Ontological relations such as "is-subclass-of" or "part-of" are used for ontological inference.

[1] Wikipedia: http://en.wikipedia.org/wiki/Ontology_(computer_science)

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 159–171, 2006.

Semantic service discovery is a service discovery process based on ontology concepts. By using ontology concepts defined in a service ontology expressively in a service description, semantics of the service description can be defined. These service descriptions are therefore expressive semantic descriptions. At the same time, by having both ontology-based descriptions and requirements, an ontology-enhanced reasoning engine (i.e. capable of ontological inference) can be used to locate services automatically and accurately. *Integrated semantic service discovery* is a semantic service discovery process based on both functional and non-functional properties of the services.

Service discovery has been a hot topic in the last years, and many different approaches have been proposed. In Web Services technology, Web Services are described in WSDL (Web Services Description Language) [22] and advertised in UDDI (Universal Description, Discovery and Integration) [21] registries. UDDI provides only keyword-based discovery (e.g. service category or provider name) and makes no use of semantic information of service behaviour (e.g. semantics of operations, inputs and outputs) defined in the service descriptions during discovery. A number of protocols for service discovery have also been proposed, most notably, SLP (Service Location Protocol) [8], Jini [11], UPnP [7] and Salutation [4]. Service descriptions in these protocols are usually based on categories of predefined service types, interface types, attributed IDs and values, without expressive semantic descriptions to enable reasoning. Thus service discovery is restricted to simple keyword-based category and attribute matching. Other approaches based on Semantic Web technology, such as [13], provide semantic service discovery, but limit their discovery to functional properties only, without considering non-functional properties during the process. Integrated semantic service discovery, however, is important to achieve accurate and satisfactory discovery results.

In this paper, we propose an integrated semantic service discovery approach based on semantic-annotated WSDL [16]. Ontologies are defined in the Web Ontology Language, OWL [12]. Behavioral semantics are added to WSDL file by associating service functionality related elements with links to OWL-based service ontology. Non-functional properties are specified as QoS parameters and rule-based policies comprising business policies, QoS policies and context policies. Furthermore, WS-Policy Framework (Web Services Policy Framework) [6] and WS-PolicyAttachment (Web Services Policy Attachment) [5] are utilized to attach QoS parameters and policies to WSDL-based service descriptions. Service requirements are also expressed using ontology concepts. Based on them, an integrated semantic service discovery procedure is presented, which takes into account selection criteria based on business policies, QoS policies and context policies, as well as user defined service selection criteria in terms of overall QoS scores based on QoS parameters.

The rest of the paper is organized as follows: Sect. 2 discusses service description elements with focus on rule-based policy specifications. Based on it, an integrated semantic service discovery framework is presented in Sect. 3. Related work is discussed in Sect. 4, followed by summary and conclusions in Sect. 5.

2 Service Description Elements

A service description comprises the following elements:

- *Functionality description* in terms of operations, inputs, outputs, preconditions and effects.
- *QoS parameters* offered by the service.
- *Service parameters* such as location and other service specific parameters.
- *Policies* for service discovery, comprising business policies, QoS policies and context policies.

A policy is a rule applied in the decision making process. It is usually conditional criteria against which factual variables are evaluated to determine an appropriate action. Therefore, a rule-based policy representation is a natural choice for policy specification. Ontology-based policy description allows service providers and requestors to describe their policies with respect to a common ontology in terms of meaningful concepts and relations. Furthermore, benefit from semantic enrichment and ontological inference can be achieved. For service discovery process, policies will guide the optimal selection of desired services among several functionally equivalent services.

Formally, an ontology-based policy rule P can be defined as a tuple <r, o, s, c, a>, where r is a reference to a policy ontology, o denotes the organization P belongs to, s denotes the service P applied to, c the conditions, a the actions. WS-Policy framework [6] provides a general purpose model to describe and communicate policies of a Web Service. It places no restrictions on the language used to represent policy expressions. We use an ontology language, OWL, to express the policy based on the upper ontology for policy inspired by [18] and depicted in Fig. 1. The upper ontology defines the concepts used for policy specification and their relations. A *policy* belongs to an *organization* and is applied to a *service.* A *policy* has a *policy domain* and refers to a domain-specific *policy ontology.* A *policy* may have multiple *rule sets*, each of them defines a set of *rules.* The *rule sets* have *rule operators*, such as "ExactlyOne" to specify that only one rule set is applied at a time. Each *rule* is a conditions-and-actions statement, which specifies the *actions* to be performed when *conditions* are evaluated to TRUE. *Conditions* are specified in *expressions* while *actions* are associated with *operations* of a *service. Actions* may also have *expressions* and may require *conditions. Expressions* may have *attributes, literal values, operators* as well as *logical operators. Attributes* can be *service parameters, inputs, outputs* or *QoS parameters* of a service. Therefore, this upper ontology of policy has relations with service ontology, i.e. concepts in the gray area in Fig. 1 also belong to service ontology.

2.1 Business Policies

Business policies are rules related to business concepts. They can be published associated with a service to constrain service discovery process. As an example, consider a delivery policy for an online bookstore, say *BookStoreA*, which specifies that if the number of copies ordered for a book is less than 50, then the

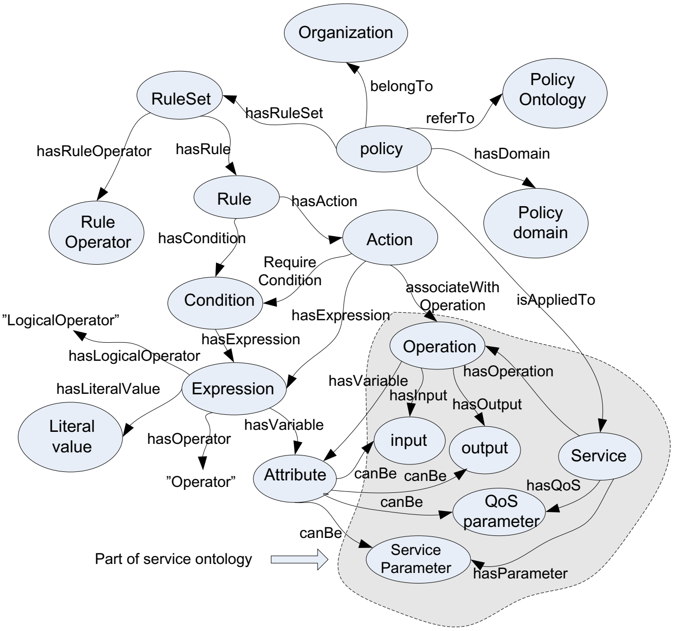


Fig. 1. Upper ontology for rule-based policy

delivery time will be within 5 days; if between 50 and 200, the order will be delivered within 10 days; otherwise, the inventory should be checked before an action is taken. The policy can be expressed in three rules as shown in Fig. 2.

Rule 1 IF ($numberOfCopies \leq 50$)
THEN DeliverBooks ($deliveryDays \leq 5$)
Rule 2 IF ($50 < numberOfCopies \leq 200$)
THEN DeliverBooks ($deliveryDays \leq 10$)
Rule 3 IF ($numberOfCopies > 200$)
THEN CheckInventoryFirst

Fig. 2. Example delivery policy for an online bookstore

A user who orders 100 copies of a textbook to be delivered within 15 days from online bookstores may select the *PurchaseBook* service from *BookStoreA* since his/her request can be matched by the second rule. Figure 3 shows part of the policy specification called *DeliveryPolicyBookStoreA* for an online bookstore *BookStoreA* based on the upper ontology defined in Fig. 1, which corresponds to **Rule 1** in Fig. 2. Note that the namespace *po:* refers to the upper ontology, while the namespace *sp:* refers to the domain-specific policy ontology. Attribute *numberOfCopies* is a service input, attribute *deliveryDays* is a service output, while operation *DeliverBookOperation* is a service operation.

```
xmlns:po=“http://examplepolicy.com/policy.owl#”
xmlns:sp=“http://ecommerce.com/policy.owl#”
xmlns=“http://BookStoreA.com/PolicyRule.owl#”

<sp:DeliveryPolicy rdf:ID=“DeliveryPolicyBookStoreA”>
  <po:hasRuleSet rdf:ID=“RuleSet1”>
    <po:hasRuleOperator rdf:resource=“po:ExactlyOne” />
    <po:hasRule>
      <po:Rule rdf:ID=“Rule1”>
        <po:hasCondition rdf:resource=“#CheckQuantity1”/>
        <po:hasAction rdf:resource=“#DeliverBooks1”/>
      </Rule>
    </po:hasRule>
    ....
  </po:hasRuleSet>
</sp:DeliveryPolicy>

<sp:CheckQuantity rdf:ID=“CheckQuantity1”>
  <po:hasExpression>
    <po:Expression rdf:ID=“ExprCondition1”>
      <po:hasVariable>
        <po:Attribute rdf:resource=“sp:numberOfCopies”/>
      </po:hasVariable>
      <po:hasOperator rdf:resource=“po:isLessThanOrEqual”/>
      <po:hasLiteralValue>
        <po:LiteralValue rdf:ID=“LiteralValue1”>
          <po:hasValue rdf:datatype=“http://www.w3.org/2001/XMLSchema#int”>50
          </po:hasValue>
          <po:hasType rdf:resource=“po:Integer” />
        </po:LiteralValue>
      </po:hasLiteralValue>
    </po:Expression>
  </po:hasExpression>
</sp:CheckQuantity>

<sp:DeliverBooks rdf:ID=“DeliverBooks1”>
  <po:associateWithOperation>
    <sp:DeliverBookOperation rdf:ID=“DelBkStoreA” />
  </po:associateWithOperation>
  <po:hasExpression>
    <po:Expression rdf:ID=“ExprAction1”>
      <po:hasVariable>
        <po:Attribute rdf:resource=“sp:deliveryDays” />
      </po:hasVariable>
      <po:hasOperator rdf:resource=“po:isLessThanOrEqual” />
      <po:hasLiteralValue>
        <po:LiteralValue rdf:ID=“LiteralValue2”>
          <po:hasValue rdf:datatype=“http://www.w3.org/2001/XMLSchema#int”>5
          </po:hasValue>
          <po:hasType rdf:resource=“po:Integer” />
        </po:LiteralValue>
      </po:hasLiteralValue>
    </po:Expression>
  </po:hasExpression>
  <po:requireCondition rdf:resource=“#CheckQuantity1”/>
</sp:DeliverBooks>
```

Fig. 3. Example policy rule specification in OWL

2.2 QoS Properties

QoS is a very important aspect of non-functional properties for a service. In a distributed environment, services with equivalent functionality can be provided by different service providers with substantially varied QoS. How to specify QoS and incorporate it into service discovery process is thus of great importance.

QoS properties can be specified as *QoS parameters* and *QoS policies*. *QoS parameters* are QoS attributes that can be expressed in quantifiable measurements or metrics. A service usually possesses a set of QoS parameters. Though many of them are of dynamic nature, i.e., related to the execution of the service, a service can still advertise its guaranteed QoS in service description. *QoS policies* are rules related to QoS parameters. Rule-based QoS policies have often been used in network-related services, e.g. network and system management services. A service can provide different QoS classes of service depending on the service classes users subscribed. For example, a *GoldClass* user may have access to *GoldClassService*, which guarantees a set of QoS parameters, such as bandwidth and response time, much better than a user in *SilverClass*. A QoS policy for service discovery can similarly be specified as "If a user belongs to *GoldClass*, then the service provided guarantees a set of QoS parameters."

QoS parameters can be classified into different categories, e.g., scalability, capacity, performance, reliability, availability, etc. There are several efforts to define and categorize QoS parameters in terms of classifications or ontologies [10] [15]. Fig. 4 shows part of a QoS ontology based on [10] for illustration purpose.

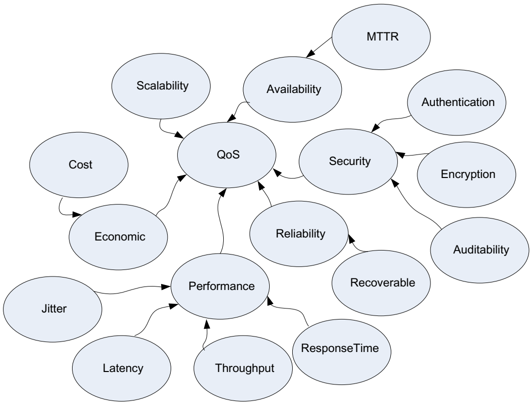


Fig. 4. Part of a QoS ontology (arrows indicate subClassOf relationship)

Not all attributes in the QoS ontology is relevant in a specific service discovery process, for example, a user may consider some of the QoS parameters valuable in his or her request. The matching procedure for service discovery therefore needs to take this into account by calculating the user specified QoS selection criteria. We adopt the QoS ranking approach proposed in [14], which defines a *quality matrix* to represent the values of user specified QoS parameters for all candidate services and an *overall QoS score function* to calculate overall QoS satisfactory values.

A *quality matrix*, $\Phi = \{V(Q_{ij}); 1 \leq i \leq m; 1 \leq j \leq n\}$, is defined as a collection of quality attribute-values for a set of candidate services, where $V(Q_{ij})$ represents the value of the i^{th} QoS attribute for the j^{th} candidate service. These values are obtained from candidate service descriptions and mapped to a scale between 0 and 1.

An *overall QoS score function* is defined as

$$f_{QoS}(Service_j) = \sum_{i=1}^{m}(V(Q_{ij}) \times Weight_i)$$

where m is the number of QoS attributes in Φ, Weight$_i$ is the weight value (specified by user) for each attribute.

The f_{QoS} score is calculated for each candidate service, and if the f_{QoS} score is greater than some user defined threshold, the corresponding service will be selected. Take an example, a user requesting an online streaming video service considers *throughput, response time* and *availability* more valuable than other QoS parameters and specifies the QoS selection criteria as Weight$_{Throughput}$ = 0.8, Weight$_{ResponseTime}$ = 0.9, Weight$_{Availability}$ = 0.7, and a threshold score value U$_{Threshold}$ = 1.5. Assume there are three candidate services for online streaming video, S_1, S_2 and S_3, and the quality matrix is:

$$\Phi = \left(\begin{array}{rccc} & S_1 & S_2 & S_3 \\ \mathit{Throughput} & 0.90 & 0.80 & 0.50 \\ \mathit{ResponseTime} & 0.90 & 0.80 & 0.60 \\ \mathit{Availability} & 0.90 & 0.50 & 0.40 \end{array}\right)$$

After calculation of their respective f_{QoS} scores, only S_1 and S_2 will be selected. Further assume that the user specifies to rank the services based on *Cost* in ascending order, and the *Cost* of S_1 is greater than that of S_2 , the results returning to the user will be $\{S_2, S_1\}$, specifying that S_2 is a better choice than S_1 for the user's purpose.

2.3 Context Policies

Context policies are rules related to context information. Some context information can greatly affect the selection of services. For instance, for a home food delivery service, the user's location is an important aspect for selecting possible service providers. In addition, depending on service types, different context information should be considered. Examples of context information include location, time, connection (e.g., if the user is accessible via a wireless or wired connection), user's feeling, presence, and user's habits and hobbies.

Context policies can be specified in the same format as business policies and QoS policies. For example, a service provider for a home food delivery service may specify a location-based policy as "only deliver food within the same city". This location policy can be specified as:

IF (*UserLocation.city = ServiceProviderLocation.city*)
THEN Service can be provided.

Some services are context-aware, others are not. For context-aware services, it is preferable that the context information can be automatically integrated into the service request even though the user does not specify them explicitly. For instance, there are several approaches to identify the location of a mobile user. One method to position the mobile user is to leverage the SS7 network to derive location. Another example is user profiles, which usually define user preferences, such as habits, hobbies, and other personalized information, such as access rights and startup applications. However, mechanisms for obtaining such context information are outside the scope of this paper. We just demand that context information should be included as a part of service request wherever possible so that context policies can be used during service discovery.

3 Integrated Semantic Service Discovery Framework

3.1 Integrated Semantic Service Description

There have been efforts to add semantics to service descriptions. Two major approaches based on ontology are OWL-S [3] and semantic-annotated WSDL [16]. OWL-S uses an OWL-based ontology for describing Web Services and supports service discovery at the semantic level. The semantic-annotated WSDL approach relates concepts in WSDL to OWL ontologies in Web Services descriptions, i.e., WSDL or UDDI. We adopt semantic-annotated WSDL to describe services, because WSDL has been accepted as the industry standard for Web Services description and most of the existing Web Services support WSDL standards. This has the advantage of having widely acceptance without adding significant complexity.

Figure 5 demonstrates the semantic annotation for our integrated service description approach. An ontology-based semantic service description is represented as a semantic-annotated WSDL file, with links to the ontology definition and WS-Policy file for policy definitions and provided QoS parameters. This semantic-annotated WSDL is an XML-formatted Web Service description document based on WSDL, and is extended with OWL-based ontologies to add semantics to WSDL elements. The WSDL file *PurchaseBookService.wsdl* specifies the functional properties in terms of operations, inputs and outputs. Preconditions and effects for the operation can also be specified [17]. The concepts of them are referred to concepts in the service ontology. Policies for service discovery are specified in OWL file *PolicyRule.owl*, while WS-Policy framework and WS-PolicyAttachment are utilized to attach them to WSDL. In order to incorporate QoS parameters into WSDL without significant changes to existing WSDL structure, QoS parameters and other service parameters are also specified in an OWL file *QoS.owl*, and attached to WSDL using the same mechanism as the policy file. In detail, this means that all policies related to the service as well as QoS and service parameters can be specified in one XML file, *Policy.xml*, with links to respective OWL files, as shown in Fig. 6. This policy specification can then be attached to WSDL description, as shown in Fig. 7. As to be noted, we assume there is a shared ontology for each service domain, the same stands for policy and QoS ontologies. At the same time, a local ontology can be extended based on shared ontology to accommodate special needs.

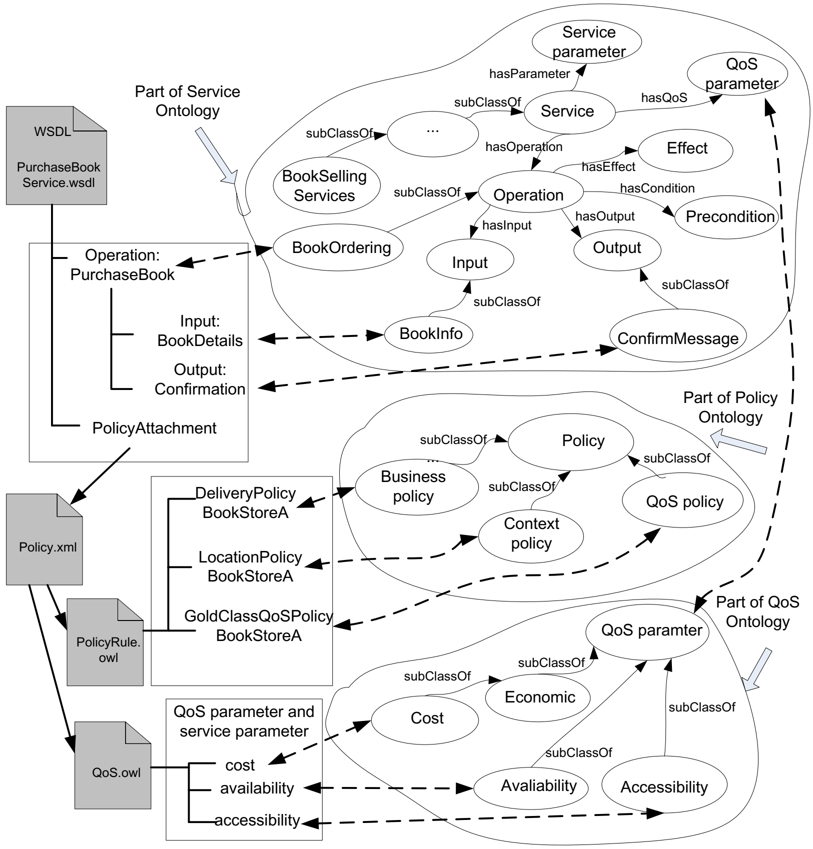


Fig. 5. Semantic annotated service description for integrated service discovery

```
<wsp:Policy Name=“PurchaseBook”>
  <wsp:All>
    <!– Business policy –>
    <po:DeliveryPolicy>http://BookStoreA.com/PolicyRule.owl#DelieveryPolicyBookStoreA
    </po:DeliveryPolicy>
    <!– Context policy –>
    <po:LocationPolicy>http://BookStoreA.com/PolicyRule.owl#LocationPolicyBookStoreA
    </po:LocationPolicy>
    <!– QoS policy –>
    <po:QoSPolicy>http://BookStoreA.com/PolicyRule.owl#GoldClassQoSPolicyBookStoreA
    </po:QoSPolicy>
    <!– QoS parameters and other service parameters –>
    <po:QoSParameters>http://BookStoreA.com/QoS.owl#QoSParametersBookStoreA
    </po:QoSParameters>
    <!– other policy for PurchaseBook, e.g. security policy –>
    ...
  </wsp:All>
</wsp:Policy>
```

Fig. 6. *Policy.xml* - All policy specification for Web Service *PurchaseBookService*

```
<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsa:EndpointReference>
      <wsa:ServiceName Name=“PurchaseBookService” />
      <wsa:PortType Name=“PurchaseBookPortType” />
      <wsa:Address URI=“http://BookStoreA.com/PurchaseBookService” />
    </wsa:EndpointReference>
  </wsp:AppliesTo>
  <wsp:PolicyReference URI=“http://BookStoreA.com/Policy.xml” />
</wsp:PolicyAttachment>
```

Fig. 7. Attaching policy specification to WSDL file *PurchaseBookService.wsdl*

3.2 Integrated Semantic Service Requirement

A user request specifies the functional requirements, non-functional requirements and user defined selection criteria in terms of preferred QoS parameters and their weights. Such request is also based on ontology concepts. A request template can be provided. This user request is combined with automatically obtained context information to produce an integrated service requirement specification in the form of an *integrated semantic service request*, which comprises the following information:

- *Functional requirements* in terms of operations, inputs, outputs, preconditions and effects.
- *Non-functional requirements*, such as QoS constraints.
- *Context information* obtained automatically by the system.
- *User specified selection criteria*, i.e. QoS parameters and their weights as well as user specified ranking criteria. This allows for personalized service ranking.

3.3 Integrated Semantic Service Discovery Procedure

Integrated semantic service discovery process can be arranged in two major steps. The first step is to find out the services that meet the functional requirements based on matching of functional properties. As there is usually more than one service matching the functional requirements, a set of candidate services are obtained. The next step is therefore to select the most appropriate ones from these candidates based on non-functional properties and rank them according to user defined criteria.

The whole discovery process is carried out by a reasoning engine. When an integrated semantic service request is sent to the reasoning engine, the engine will first determine the candidate services that offer the requested functionality based on matching of functional properties. We adopt a procedure based on ontological inference and degree of match [13]. This procedure typically uses subsumption reasoning to find similarity between service descriptions and service requests based on operations, inputs and outputs. Preconditions and effects can also be used for matching. During the second step, policies will be checked and applied to further select services among the candidate services. The semantic-annotated WSDL files of candidate services contain links to all policy specification file (e.g. *Policy.xml*), which can be referenced to retrieve the related policy rules (*PolicyRule.owl*) as well as QoS and service parameters (*QoS.owl*). Rule-based reasoning can then be applied to determine satisfied matching. After that, overall QoS scores for those candidate services are calculated based on user defined selection criteria (i.e. based on selected QoS parameters and their respective weights) as described in Sect.2.2. All the matches will be returned according to user specified ranking criteria.

Ontological inference and rule-based reasoning are applied during semantic service discovery process. The reasoning engine which carries out the above procedure is based on XDD [23] - a knowledge representation framework - and XET [2] - a powerful computing and reasoning engine for XDD. Work has already been done, based on this representation framework and reasoning engine, for dynamic service configuration [1], composition [20], and management [9], proving the practicability and reasoning power of such reasoning engine.

XDD (XML Declarative Description) is an expressive XML rule-based knowledge representation, which extends ordinary, well-formed XML elements by incorporation of variables for an enhancement of expressive power and representation of information into so called *XML expressions*. A description in XDD is a set of XML expressions and the XML elements' relationships in terms of *XML clauses*. XML expressions represent facts, while XML clauses express rules, conditional relationships, constraints and ontological axioms. Applying XDD framework, Ontology-annotated WSDL descriptions, concepts and properties in OWL-based ontologies, service parameters and QoS parameters can all be represented as facts using XML unit clauses. Ontological relations and axioms as well as policy rules for service discovery can be represented as rules using XML non-unit clauses. Rules can also be defined for ontological inference and querying in XDD. Service requests can be represented as XDD query clauses using XML clauses, which specify the patterns as well as the selection conditions of the queries. This means all information and rules for integrated semantic service discovery can be directly represented as XDD descriptions. Furthermore, XDD descriptions can be computed and reasoned using XET (XML Equivalent Transformation), a Java-based reasoning engine that transforms the query clause by the XDD-based rules based on equivalent transformation [2]. Therefore, by expressing ontologies and rules directly in XDD and executing service discovery queries using XET-based engines, we eliminate the overhead of transforming between ontology language and rule-based representation, and can achieve both ontological inference and rule-based reasoning, two fundamental functionalities for semantic service discovery process.

4 Related Work

Several approaches for ontology-based semantic service discovery have been proposed, based on OWL-S [13] or semantic-annotated WSDL [16]. However, both of them only apply ontology for matching on the operational interfaces (i.e. input and output parameters of the operations of the Web Services). In addition, both of them lack mechanisms to represent non-functional properties based on rule-based policies. We extend the semantic matching and selection based on non-functional properties, i.e. ontology-based policy rules and QoS parameters.

Sriharee et al. [18] proposed to discover Web Services based on business rules policy using WS-Policy and ontology, but without further consideration of QoS attributes. For incorporating QoS attributes with service discovery, Zhou et al. [24] proposed a DAML-QoS ontology for specifying various QoS properties and metrics. However, there was no provision for the users to specify ranking criteria (based on non-functional properties) for service selection. The framework proposed by Pathak et al. [14] provides QoS-based service selection; however, there was no consideration for policy rules during the discovery process. Maximilien et al. [10] proposed a framework and ontology for service selection also considering QoS properties, but there was no provision for user-specified ranking criteria in service request.

5 Conclusions

An approach to integrated semantic service discovery is presented. We first describe how non-functional properties are expressed based on business policies, QoS policies and context policies as well as QoS parameters. We then present our approach for adding semantics to service description for both functional and non-functional properties based on ontologies. We further show how service request can be integrated with context information and personalized ranking criteria. Based on them, an integrated semantic service discovery procedure based on both functional and non-functional properties is presented.

We base our work on widely accepted standards in Web Services and Semantic Web, i.e., WSDL, OWL and WS-Policy. The integrated semantic service discovery approach is a rather generic one, and can be applied in a centralized or distributed environment. We are working towards mechanisms to apply this approach to autonomic environments with distributed, self-organizing and scale-free communications. Issues about how the service descriptions are stored and organized as well as how they are accessed need further exploration.

Shared ontologies are assumed for service descriptions and service requests in our approach. If different ontologies are used, ontology mapping should be carried out to build up correspondence between ontologies used for service descriptions and those used for service requests.

References

1. F. A. Aagesen, P. Supadulchai, C. Anutariya, and M. M. Shiaa. Configuration management for an adaptable service system. In *IFIP Int'l Conference on Metropolitan Area Networks, Architecture, Protocols, Control and Management, proceedings*, Ho ChiMinh City, VietNam, 2005.
2. C. Anutariya, V. Wuwongse, and V. Wattanapailin. An equivalent-transformation-based xml rule language. In *Int'l Workshop Rule Markup Languages for Business Rules in the Semantic Web, proceedings*, Sardinia, Italy, 2002.

3. The OWL Services Coalition. Owl-s: Semantic markup for web services, 2003. http://www.daml.org/services/owl-s/1.0/owl-s.html.
4. The Salutation Consortium. Salutation architecture specification version 2.0c, 1999. http://www.salutation.org/.
5. S. Bajaj et al. Web services policy attachment, 2006. http://www-128.ibm.com/developerworks/library/specification/ws-polatt/.
6. S. Bajaj et al. Web services policy framework (ws-policy), 2006. http://www-128.ibm.com/developerworks/library/specification/ws-polfram/.
7. UPnP Forum. Upnp device architecture version 1.0, 2000. http://www.upnp.org/.
8. E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2. RFC2608, 1999.
9. S. Jiang, M. M. Shiaa, and F. A. Aagesen. An approach for dynamic service management. In *EUNICE'04, Proceedings*, Tampere, Finland, 2004.
10. E. M. Maximilien and M. P. Singh. A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8(5):84–93, 2004.
11. Sun Microsystems. Jini architecture specification version 2.0, 2003. http://www.jini.org/.
12. OWL. Owl web ontology language overview. W3C Recommendation, Feb 2004. http://www.w3.org/TR/owl-features/.
13. M. Paolucci, T. Kawmura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In *First Int. Semantic Web Conf., Proceedings*, 2002.
14. J. Pathak, N. Koul, D. Caragea, and V. Honavar. A framework for semantic web services discovery. In *WIDM'05, Proceedings*, 2005.
15. S. Ran. A model for web services discovery with qos. *ACM SIGecom Exchanges*, 4(1):1–10, 2003.
16. K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller. Adding semantics to web services standards. In *ICWS'03, Proceedings*, 2003.
17. N. Sriharee and T. Senivongse. Discovering web services using behavioural constraints and ontology. In *DAIS'03*, volume 2893 of *LNCS*, pages 248–259. Springer, 2003.
18. N. Sriharee, T. Senivongse, K. Verma, and S. Sheth. On using ws-policy, ontology, and rule reasoning to discover web services. In *INTELLCOMM 2004, Proceedings*, volume 3283 of *LNCS*, pages 246–255, Bangkok, Thailand, 2004. Springer.
19. R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197, 1998.
20. P. Supadulchai and F. A. Aagesen. A framework for dynamic service composition. In *First Int'l IEEE Workshop on Autonomic Communications and Computing, Proceedings*, Taormina, Italy, 2005.
21. uddi.org. Universal description, discovery and integration of web services. http://www.uddi.org/.
22. W3C. Web services description language (wsdl)1.1, 2001. http://www.w3.org/TR/wsdl.
23. V. Wuwongse, C. Anutariya, K. Akama, and E. Natajeewarawat. Xml declarative description: A language for the semantic web. *IEEE Intelligent Systems*, 16(3): 54–65, 2001.
24. C. Zhou, L. Chia, and B. Lee. Service discovery and measurement based on daml-qos ontology. In *Special Interest Tracks and Posters of 14th World Wide Web Conference, Proceedings*, 2005.

Policy-Based Management and Context Modelling Contributions for Supporting Services in Autonomic Systems

J. Martín Serrano[1], Joan Serrat[1], John Strassner[2], and Ray Carroll[3]

[1] Universitat Politècnica de Catalunya. Barcelona, Spain
{jmserrano; serrat}@tsc.upc.edu
[2] Motorola Labs, Schaumburg, IL, USA
{john.strassner@motorola.com}
[3] Telecommunications, Systems and Software Group,
Waterford Institute of Technology. Waterford, Ireland
{rcarroll@tssg.org}

Abstract. Autonomic networking systems dynamically adapt the services and resources that they provide to meet the changing needs of users and/or in response to changing environmental conditions. This paper presents a novel policy-based, context-aware, service management framework for ensuring the efficient delivery and management of next generation services using autonomic computing principles. The novelty of this approach is its use of contextual information to drive policy-based changes that adapt network services and resources. Policies control the deployment and management of services and resources, as well as the software used to create, manage, and destroy these services and resources. Policies also control the distribution and deployment of the necessary components for fully managing the service lifecycle, and provide the efficiency and scalability necessary for supporting autonomic systems.

Keywords: Autonomic Systems, Autonomic Networking, Policy-Based Service Management, Context Information, Self-Management Technologies, Next Generation Services, Context-Aware Framework, Context Information Model.

1 Introduction

Over the last decade, simple advances in resources and services have been feasible as result of the ever-increasing power and growth of technologies. However, this drive for more functionality has dramatically increased the complexity of systems – so much so that it is now impossible for a human to visualise, much less manage, all of the different operational scenarios that are possible in today's complex systems. The stovepipe systems that are currently common in OSS and BSS design exemplify this – their desire to incorporate best of breed functionality prohibits the sharing and reuse of common data, and point out the inability of current management systems to address the increase in operational, system, and business complexity [1]. Operational and system complexity are spurred on by the exploitation of increases in technology to build more functionality. The price that has been paid is the increased complexity of

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 172 – 187, 2006.

system installation, maintenance, (re)configuration and tuning, complicating the administration and usage of the system. Business complexity is also increasing, with end-users wanting more functionality and more simplicity. This requires an increase in intelligence in the system, which defines the need for autonomic networking [2]. If autonomic computing, as described in [1][3] is to be realized, then the ***needs of the business must be able to drive the services and resources that the network provides***.

Autonomic systems were built to manage the increasing complexity of systems [3][4] as well as to dynamically respond to changes in the managed environment [1]. While many have proposed some variant of automatic code generation, this requires a detailed model of the system that is being reconfigured as well as the surrounding environment. More importantly, a system component can not be reconfigured if the system does not understand the functionality of the component. This means that the system requires ***self-knowledge*** of its component, and knowledge of its users and requirements. One of the most important forms of knowledge is ***context***. For a system to be able to generate code to modify itself, a model of *context* and its relation to not just the set of self-configuration, self-healing, and other operations, but in general to the management environment itself, is required. In addition, policy management is in theory able to control these functions and ensure that they are applied consistently.

There are important business and technical drivers encouraging the use of autonomic principles [3]. Our work in autonomic systems centers on using the combination of policy management and context information in a formal and efficient way for supporting and managing services. Formal indicates that the specification of context should be depicted through a representational language; efficient means that context are gathered and distributed in many layers (e.g., customer site as well as the core network), and hence requires *semantics* to relate them. This paper presents a novel policy-based service management framework for ensuring the efficient delivery and management of next generation services using autonomic principles. We present an architecture that uses an innovative *fusion* of semantic information that relates the current context to services and resources delivered by the network. Different self-functions, such as self-configuration, are controlled by policy-based management. From a service oriented architecture viewpoint, policies are used for distributing and deploying the necessary services, either atomically or through composition.

This paper provides an overview of our approach. In section II we provide an overview of policy based management, and how it relates to autonomic systems. Section III describes the functionality of the policy-based service management framework, including its corresponding functional blocks and their mutual relationships. It also introduces the "policy-based paradigm" to service management, which is a novel framework for supporting service management functionality in autonomic systems. Section IV presents the policy model that is currently implemented and deployed as our approach to providing next generation services. Section V presents the validation and results for our policy-based management system. Section VI presents the conclusions, and finally the acknowledgements and bibliography references are included.

2 Policy-Based Paradigm

Policy based management has been proven as a useful paradigm in the area of network management. In the last few years, several initiatives have used policy management approaches to tackle the problem of fast and customisable service delivery. These include OPES [5] and E-Services [6]. We go one step further and present an architecture that is intended to control the full service life cycle by means of policies; in addition, it takes into account the variation in context information, and relates those variations to changes in the services operation and performance. The synergy obtained from the autonomic systems and the policy-based paradigm is the knowledge platform, which is another innovative aspect of our work.

A policy has been defined in the sense of administrator-specified directives that manages and provides guidelines for the different network and service elements in [7]. In other words, a policy is a directive that is administratively specified to manage certain aspects of desirable or needed behaviour resulting from the interactions of user, applications and existing resources [8]. In this paper, we use the definition "Policy is a set of rules that are used to manage and control the changing and/or maintaining of the state of one or more managed objects" [4]. The inclusion of *state* is very important for autonomic systems, as state is the means by which we know if our goals have been achieved, and if the changes that are being made are helping or not.

The main benefits from using policies are improved scalability and flexibility for managing services. *Flexibility* is achieved by separating the policy from the implementation of the managed service, while *scalability* is improved by uniformly applying the same policy to large sets of devices and objects. Policy-based management emerged in the network management community and it is supported by standards organisations such as the IETF, DMTF, and TMF [9][10][11]. In next generation network (NGN) usage, the application of policies is being abstracted to facilitate the works of service customisation, creation, definition and management. Another benefit from using policies when managing services is their *simplicity*. This simplicity is achieved by means of two basic techniques: centralised configuration (e.g., each element does not have to be configured individually), and simplified abstraction (e.g., each device does not have to be explicitly and manually configured – rather, a set of policies is established that *governs* desired behaviour, and the system will translate this policy into device-specific commands and enforce its correct implementation.

The main objective of using policies for service management is the same of managing networks with policies: we want to automate management and do it using as high a level of abstraction as possible. The philosophy for managing a resource, a network or a service with a policy-based managed approach is that "IF" something happens "THEN" the management system is going to take an action. The main idea is to use generic policies that can be customised following user subscription; the parameters of the conditions and actions in the policies are different for each user, reflecting its personal characteristics and its desired context information. We use the policy-based paradigm to express the service life-cycle and subsequently manage its configuration in a dynamic manner. It is this characteristic which provides the necessary support and operations of autonomic systems. It should be noted that [4] and [11] propose an important extension and enhancement to the simpler definitions employed by the IETF and DMTF that is very attractive to autonomic systems. Specifically, the definition of policies is linked specific to state management. The following definitions are from [1] and will be used in our work:

"Policy is a set of rules that are used to manage and control the changing and/or maintaining of the state of one or more managed objects."

"A PolicyRule is an intelligent container. It contains data that define how the PolicyRule is used in a managed environment as well as a specification of behavior that dictates how the managed entities that it applies to will interact. The contained data is of four types: (1) data and metadata that define the semantics and behavior of the policy rule and the behavior that it imposes on the rest of the system, (2) a set of events that can be used to trigger evaluation of condition clause of a policy rule, (3) an aggregated set of policy conditions, and (4) an aggregated set of policy actions."

Policy management is best expressed using a language. However, there are multiple constituencies involved (e.g., business people, architects, programmers, and technicians). It is shown in Figure 1. While most of these constituencies would like to use some form of restricted natural language, this desire becomes much more important for the business and end users. This notion was codified as the Policy Continuum in [4][11]. Our approach is based on producing a formal language (with appropriate dialects matched to each constituency) for standardisation in near future.

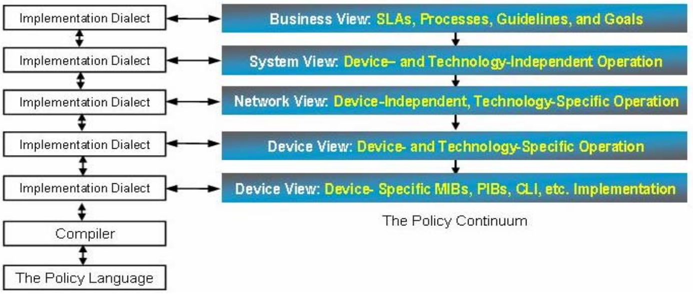

Implementation Dialect
Business View: SLAs, Processes, Guidelines, and Goals
Implementation Dialect
System View: Device– and Technology-Independent Operation
Implementation Dialect
Network View: Device-Independent, Technology-Specific Operation
Implementation Dialect
Device View: Device- and Technology-Specific Operation
Implementation Dialect
Device View: Device- Specific MIBs, PIBs, CLI, etc. Implementation
Compiler
The Policy Language
The Policy Continuum

Fig. 1. Mapping of Policy Languages to the Policy Continuum

Our approach defines this set of languages in XML to ensure platform independence. It is also easy to understand and manage, and the large variety of off-the-shelf tools and freely available software provides powerful and cost-effective editing and processing capabilities. Each of the implementation dialects shown in Figure 1 is derived by successively removing vocabulary and grammar from the full policy language to make the dialect suitable for the appropriate level in the Policy Continuum. For some levels, vocabulary substitution using ontologies is used to enable more intuitive GUIs to be built, but this is beyond the scope of this paper.

The policies are used in the managing of various aspects of the services lifecycle, an important aspect of policy-based service management is the deployment of services throughout the programmable elements. For instance, when a service is going to be deployed over any type of network, decisions that have to be taken in order to determine in which network elements the service is going to be installed and/or supported by. This is most effectively done through the use of policies that map the user and his or her desired context to the capabilities of the set of networks that are going to support the service. Moreover, service invocation and execution can also be controlled by policies, which enable a flexible approach for customising one or more service *templates* to multiple users. Furthermore, the maintenance of the code realising the service, as well as the assurance of the service, can all be related using policies. Additionally when some variations in the service are sensed by the system, one or more policies can define what actions need to be taken to solve the problem.

There are important approaches and projects dealing with Policy Based Service Management (PBSM). A survey of policy specification approaches has been provided [12]. The ANDROID Project [13] aims to prove the feasibility of providing a managed, scalable, programmable network infrastructure. A more service layer oriented approach is a proposed project [14]. The overall objective of the TEQUILA project [15] is to study, specify, implement and validate a set of service definition and traffic engineering tools to obtain quantitative end-to-end quality of service guarantees through careful planning, dimensioning and dynamic control of scalable and simple qualitative traffic management techniques within Internet services. Another interesting proposal is the CONTEXT project [16], the objective is to provide a solution in the form of an architecture that is oriented for creating, deploying and managing context-aware services. This approach is well oriented under the scope of policy based systems, but was developed without considering the multiplicity of context information and the technology non-dependence. Generalised policy-based service management architecture for autonomic systems will articulate a functional process model and include process inter-relationships for an organisation. It shall encompass a methodology that identifies the necessary policies, practices, procedures, guidelines, standards, conventions, and rules needed to support the business and their process inter-relationship enabling the organisation to govern the application of policy management mechanisms. Hence, application of Policy-Based paradigm in the service management area seems to be a feasible alternative for meeting next generation service management goals.

3 Policy-Based Service Management Framework Approach

The PBSM framework that we present deals with the creation and modelling of services, including their context information, in order to provide service assurance and management functions. It does this by constructing a framework using autonomic elements for the management and execution of services. The framework uses programmable networks [17] implemented using autonomic elements to provide the suitable execution environment for the services [2].

The framework allows the use of appropriate APIs for deploying the services in mobile and IP domains. The context-policy model that we propose is flexible enough to accommodate the value of context information that needs to be evaluated, especially if the context changes [18]. Our context-policy model overcomes many of the approaches done in the area of policy-based service management. It is based on standards as much as possible and moreover, the policy model scales with the network or applications that use it. Our context-policy model is expected to be accessible from autonomic elements that can reside in heterogeneous networks and architectures.

Storage and retrieval of this information is also important; we meet this requirement by being defined and implemented using an object-oriented philosophy.

The architecture deal with autonomic systems, for instance as an external management method such as a finite state machine, containing the desired state of the system. The autonomic system then compares the *current* state to the *desired* state and, if different, orchestrates any required changes (e.g., reconfiguration). The referred autonomic part is shown in Fig. 2.

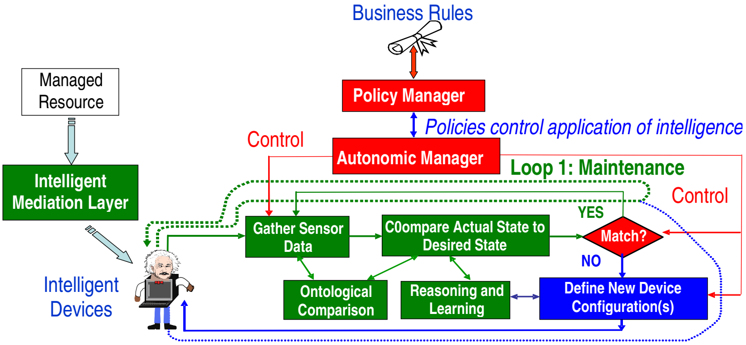


Fig. 2. Behavioral Orchestration Using an Autonomic System and Policy Management

The Intelligent Mediation Layer is a set of software that enables existing (i.e., legacy) and future (i.e., autonomic-enabled) entities to be governed by the autonomic system. Since network devices can have different programming languages and types of management information, we use DEN-ng to provide a Network *lingua franca* (i.e., a controlled vocabulary, along with a consistent set of meanings) for all management information. The "Gather Sensor Data" is responsible for harmonising data received from different sources. The "Ontological Comparison" describes *concepts* and relationships between objects defined in the DEN-ng information model, enabling the Autonomic Manager to *reason* about data received and *infer* meaning. For example, we can find out which customers are affected by a particular SNMP alarm, even though this information is not contained in the SNMP alarm, by looking at relationships in both the information model and relevant ontologies and deducing which customers are affected by the SNMP alarm. The autonomic system then determines whether the actual state of the managed entity is equal to its desired state. If it is, then the system keeps on monitoring the entity. If it is not, then the system defines the set of configuration changes that should be sent to the managed entity. These are translated by the Intelligent Middleware into vendor-specific commands for legacy as well as future devices. The Autonomic Manager provides the novel ability to change the different control functions (such as what data to gather and how to gather it) to best suit the needs of the changing environment. An important, but future, area of research is to add learning and reasoning algorithms to the system. Finally, the Policy Manager is both an interface to the outside world (GUI and/or scripted) as well as the translation of those requests to the autonomic system.

3.1 PBSM Framework Functional Block Description

The framework is composed of four main functional blocks as shown in Fig. 3. The Code Distributor is intended to download the generated code to the appropriate code repositories and/or storage points (it could also be downloaded directly to entities that can reconfigure themselves using this mechanism; this is part of our future work). This installation process is a precondition for a service to be delivered. This is because the autonomic system functions by dynamically adapting the services and resources that they deliver, and one way of providing new or different functionality is through self-configuration. In particular, our approach uses a variant of the model driven architecture [19] to achieve this.

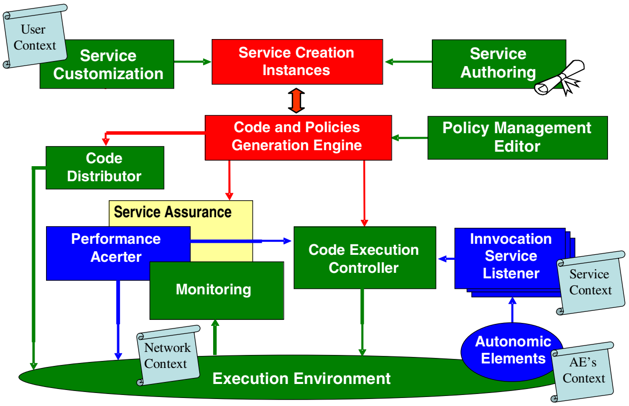


Fig. 3. Policy-Based Service Management framework functional components

Code repositories will be distributed throughout the network infrastructure. Therefore, the Distributor will need to keep track of the URIs where code was installed. The Code Execution Controller is the functional entity that will download the code corresponding to the desired service from one of the repositories to install it in the service execution environment and subsequently activate the service. The Invocation Service Listener captures any of the different types of triggers (e.g., an event, such as a user logging onto the network, or the change in state of a managed entity) that may cause this service to be activated. This component also is responsible for communicating these events to the autonomic elements that provide these services. Finally, the Service Assurance module continuously monitors the quality of service. The usual sequence of events would start with the detection of a trigger (perhaps caused by an explicit or implicit consumer request) by the Invocation Service Listener that detects a problem with the service; the service listener binds the entities shown in Fig. 3 to the appropriate autonomic elements and informs the Code Execution Controller about this problem. The Code Execution Controller would compose and forward an activation request message to the appropriate programmable nodes in the service execution environment, which in turn would retrieve code from the repository and execute it, starting the service provisioning phase. As soon as the service has been started, Service Assurance module would keep track of its behavior.

3.2 PBSM Framework Components Description

The PBSM framework is composed of a set of components as shown in Fig. 4. The key component is the *Policy Manager*, which supports and governs the functionality of all of the remaining modules *including* the Policy's Consumers. The *Authorisation Check Component* verifies that the user introducing the policy has the necessary access rights to perform this function. This is supported using the role-based access control model of DEN-ng. The *Policy Conflict Check* component is responsible for maintaining the consistency of all policies introduced into the system. E.g. each dialect of the policy language has its own vocabulary and grammar rules. The system will check a policy against these rules to ensure that the policy is well-formed. Decisions of when a policy must be enforced are closely linked with policy's Condition Evaluators, which are coordinated by the Decision Making Component.

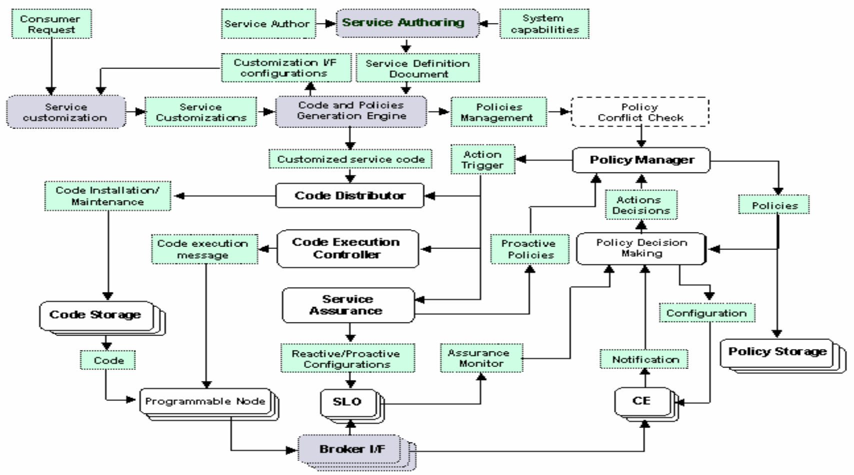


Fig. 4. Policy-Based Service Management framework Components and Interfaces

For example the Decision Making Component checks its list of business rules that must be enforced against the list of submitted policies, and notifies the administrator of any matches. It also provides valuable information for policy conflict detection and resolution. Finally, the *Policy Repository* (Database) gives support to all other components by storing Information Model objects that provide a picture of the system status and the Policies ruling the system at any time. Note that this makes use of the DEN-ng policy model [4], which is an ordered event-condition-action triplet. That is, WHEN an event occurs, IF a set of conditions are satisfied, THEN a set of actions are executed. Policy components are defined and stored as Information Model objects. Using the DEN-ng approach [4][11] the policy model is instantiated.

3.2.1 Policy Manager (PM)

The Policy Manager (PM) has two main functions: 1) to govern the functionality of the system components, and 2) to check and incorporate new policies that arrive from a Policy Generation Engine. The specific fields required by these policies and its general structure are explained in the section "Policy Model". Once a new policy, or policies, arrives as an XML document from the Policy Generation Engine, the PM parses it into Java and sends it to the Policy Conflict Check Component, which checks for conflicts with other policies that are already loaded in the system. This provides the flexibility to a) reject the policy if it conflicts with an already deployed policy, b) replace an already deployed policy with this new policy (which of course involves locating and gracefully stopping policy execution of the previous instance), or c) note that there is an unresolved conflict and wait for a human administrator to resolve it.

The PM uses a sequential algorithm, aided by the atomic characteristics from the policies (described in the policy model), to decide which policies will be activated at any given moment, otherwise it will be stored in the repository for further processing. This is governed by several factors in the DEN-ng policy model [4]. Briefly put, two types of containment semantics are provided by DEN-ng for PolicyRules: grouping and nesting. Grouping is the simpler of the two methods. It has the semantics of an assembly of PolicyRules. The PolicyGroup establishes its own hierarchy for evaluation purposes, and all immediate children (whether PolicyGroups or PolicyRules) that are directly contained within a given PolicyGroup are treated as being at the same level of containment. Finally, the DEN-ng model provides a set of metadata attributes that control the semantics of a PolicyRule or PolicyGroup. The decisionStrategy attribute is used to determine whether a single or potentially a group of PolicyRules should be evaluated, and how it is done. The isMandatory attribute defines whether a rule must be executed, and hence should be used with care, since it requires the evaluation of the condition clause of a PolicyRule and the execution of the actions of a PolicyRule if the condition clause evaluates to TRUE.

3.2.2 Decision Making Component (DMC)

The DMC receives the policy conditions from the PM. When the DMC receives the conditions of a particular policy, it determines when this condition became true and notifies the PM when the condition is met. The PM then notifies the appropriate consumers of the policy that the policy actions are ready to be executed. During the evaluation process three main activities must be performed: obtain the values of the *Condition Objects*; evaluate the *Condition Requirements*; and apply the *Evaluation Method*. The DMC will use the components called Condition Evaluators (CEs) in order to support this evaluation process that deal with the autonomic elements.

3.2.3 Policy and Information Model Repository

The Policy & Information Model Repository is the logical place where policies and other needed management information will be stored and fetched when needed. The Repository will store information about the Policies loaded in the system, information about the network, the available components (CE & AC) and their capabilities (variables they can monitor and actions they can enforce), and other information entities (e.g. management information). This information is of great importance, for example when evaluating conditions, in order to determine whether the appropriate Condition

Evaluators are already installed or not to monitor appropriate variables. The Policy Definition System will have to be aware of the different Action Consumer and Condition Evaluator components, as well as the different actions and conditions they are able to enforce and evaluate, in order to effectively use and activate Management Policies.

3.2.4 Policy Conflict Resolution (PCR)

The Policy Conflict Resolution component (PCR) has the responsibility of maintaining the overall consistency of the system. One example of inconsistency would be that two policies require opposite actions when the same conditions are met. The functionality of the PCR is to detect these situations and only accept new policies when they do not introduce inconsistencies into the system. The algorithms required to detect inconsistencies are not straightforward, and an active area of ongoing research. For this reason the implementation of this component is not yet done, and the component has been mentioned only for completeness.

3.2.5 Action Consumers (AC)

The Action Consumers (ACs) are the components for enforcing the policy actions after the request of the PM. The PM will send to the appropriate Action Consumer the action to be enforced and the initial parameters (if needed) to be used in order to do that. The AC will return the results of the enforcement, showing if the action has been successfully enforced. The ACs should be dynamically installed when they are needed. To support this feature, it is necessary that the code(s) of the ACs are stored in one (or more) Code Storage Points, playing the role of a Management Code Repository. An example of one API for these components is given below:

```
public class CodeInstallerInterface {
    public CodeInstallerInterface();
    public boolean InstallCode(String CodeId, URL[] URLList, int NoCopies, int Level, String[] PotentialExecPoints);
    public URL[] GetOptimalURLOfCode(String CodeId, InetAddress DINANodeIPAddr);
    public boolean RemoveCode(String CodeId);
}
```

3.2.6 Condition Evaluators (CE)

The CEs are software components that can be installed in different parts of the network or network components. They interact with the appropriate autonomic elements to get the values required by a given condition. The CEs perform monitoring and requirement evaluation activities, and can be responsible for evaluating requirements (apply Condition Requirements) if all the events or variables (Condition Objects) needed to evaluate the requirements are obtained. The Condition Object is a PolicyStatement [4] that has the generic form {variable operator value}. DEN-ng enables variables and values to be modeled as classes to facilitate reuse; runtime substitution (e.g., with a literal) is also supported. An example of one API for these components is given as follows:

```
public class userPositionController {
    public userPositionController ();
    public int userPosition(String userID, String X, String Y, String Z, String [] arg);
    public int userPosition(String userID, String X, String Y, String Z, String mapReference, String [] arg);
    public int sendMessageToService(String user ID, String X, String Y, String Z, String msg);
    public int sendMessageToService(String userID,  String X, String Y, String Z, String mapReference, String msg);
}
```

4 Policy Model

The policies are structured hierarchically, in terms of Policy Sets, which can be either PolicyRules or Policy Groups. The Policy Groups can contain PolicyRules and/or other Policy Groups. This is enabled through the use of the composite pattern for defining a PolicySet, and is shown in Fig. 5. That is, a PolicySet is defined as either a PolicyGroup or a PolicyRule. The aggregation HasPolicySets means that a PolicyGroup can contain zero or more PolicySets, which in turn means that a PolicyGroup can contain a PolicyGroup and/or a PolicyRule. In this way, hierarchies of PolicyGroups can be defined.

The order of execution of PolicyRules and PolicyGroups depends on the structure of the hierarchy (e.g., grouped and/or nested) and is controlled by the set of metadata attributes defined in Section 0. The high level description of policies follows the following format:

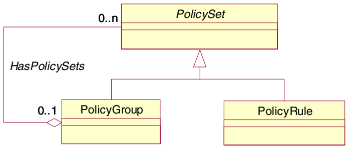


Fig. 5. The Composite Pattern Applied to the Defintion of PolicySets

WHEN an event_clause is received that triggers a condition_clause evaluation
IF a condition_clause evaluates to TRUE, subject to the evaluation strategy
THEN execute one or more actions, subject to the rule execution strategy

The above has the following semantics. Policies are not evaluated until an event that triggers their evaluation is processed. Note that the event does not have to be actively sent to the autonomic system by a network component (e.g. as in an SNMP alarm) – passive events, such as the passing of time, can also be included. The Service Management Policies control just the service lifecycle, never the logic of the service. In this way, Service Management policies are used by the PBSM components of the system to define the Code Distribution and Code Maintenance of the service as well as the Service Invocation, Service Execution and Service Assurance processes. Coming from the general requirements associated to the Service Management layer, we have defined five types of policies covering the service lifecycle. These five policy types are structured around an information model whose most representative part is shown in Fig. 6.

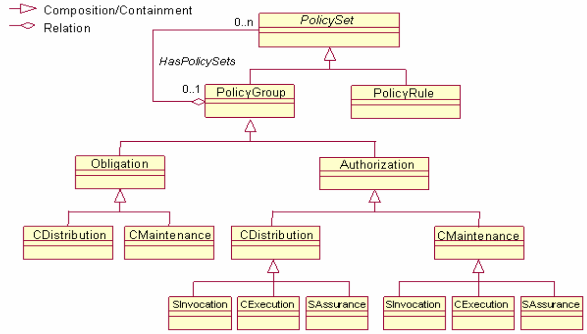


Fig. 6. Policy Information Model Hierarchy

We have extended the above model by adding the new concept *atomicity* of a PolicyGroup or PolicyRule. If the first PolicyRule or PolicyGroup is labeled as Atomic, then the second PolicyRule or PolicyGroup will be activated only if the first one has been previously enforced. If it is not then the second Policy can be activated simultaneously with the first one. Note that if a PolicyGroup is labelled as Atomic, then all the Policies contained have to be previously enforced before *any* Policies of the second PolicyGroup can be activated. The different Policy Groups inside a Policy Set or inside other Policy Groups are also ordered with a sequence number. The first Policy Group in the sequence will be processed first, then the next, and so forth.

4.1 Service Code Distribution

This step takes place immediately after the service creation and customisation. It consists of storing the service code in specific storage points. Policies controlling this phase are *CDistribution Policies*. The mechanism controlling the code distribution will determine in which storage points the code is to be stored. The enforcement will be carried out by the Code Distribution Action Consumer. A high level example of this type of policies is presented:

"**If** (customised service B code is received)
then (configure distribution of service B code and optimum Storage Point selection parameters)"

4.2 Service Code Maintenance

Once the code is distributed, it must be maintained in order to support updates and new versions. For this task, we have the *CMaintenance Policies*. These policies control the maintenance activities carried out by the system on the code of specific services. A typical trigger for these policies could be the creation of a new code version or the usage of a service by the consumer. The actions include code removal, update and redistribution. These policies will be enforced by the Code Distribution Action Consumer. Three high level examples of this type of policies are shown here:

"**If** (new version of customised service B code)
then (remove old code version of service B from Storage Points) & (distribute new service B code)"

"**If** (customised service B code expiration date has been reached)
then (deactivate execution polices for service B) & (remove code of service B from Storage Points)"

"**If** (The invocation's number for service B is very high)
then (distribute more service B code replicas to new Storage Points)"

4.3 Service Invocation

The service invocation is controlled by *SInvocation Policies.* The Service Invocation Condition Evaluator detects specific triggers produced by the service consumers. These triggers also contain the necessary information that the policy is going to evaluate in order to determine the associated actions such as addressing a specific code repository and/or sending the code to specific execution environments in the network. The policy enforcement takes place in the Code Execution Controller Action Consumer. A high level example of this type of policies is presented:

```
"If (invocation event X is received)
    then (customised service B must be downloaded to the specific IP address) "
```

4.4 Code Execution

CExecution Policies will drive how the service code is executed. This means that the decision about where to execute the service code is based on one or more factors (e.g., using performance data monitored from different network nodes, or based on one or more context parameters, such as location or user identity). Service Assurance Action Consumers are entrusted to evaluate such network conditions and the enforcement will be the responsibility of the Code Execution Controller Action Consumer. A high level example of this type of policies is presented:

```
"If (invocation event X is received)
   then (customised service B must be executed)"
```

4.5 Service Assurance

This phase is under the control of *SAssurance Policies, which* are intended to specify the system behaviour under service quality violations. Rule conditions are evaluated by the Service Assurance Condition Evaluator. These policies include preventive or proactive actions, which will be enforced by the Service Assurance Action Consumer. Two high level examples of this type of policies are presented:

```
"If (customised service B is running)
   then (configure assurance parameters for service B) &  (configure local assurance variables)"
```

```
"If (level=2)&(parameterA>X) then (Action M)"
"If (level=2)&(parameterB>Y) then (Action N)"
"If (level=2)&(parameterC<Z) then (level=1)&(Action K)"
```

The proposed PBSM framework does not assume a 'static' information model (i.e., a particular, well defined vocabulary that does not change) for expressing policies. In contrast, our proposed framework can process policies that can be defined dynamically (e.g., new variable classes can be defined at run-time). The essence of our framework approach is to associate the information expressing policy structure, conditions and actions with information coming from the external environment. Specifically, the externally provided information can either match pre-defined schema elements or, more importantly, can extend these schema elements. The extension requires machine-based reasoning to determine the semantics and relationships between the new data and the previously modelled data. This is new work, an overview of which is in [1], and is beyond the scope of this paper. Information consistency and completeness is guaranteed by a policy-definition system, which is assumed to reside outside the proposed framework (the service creation and customisation systems in the Context system).

By supporting dynamically defined policies, we achieve the flexibility of policy-based management. We think that this feature is indeed a requirement of the design of the overall Context system (for achieving rapid context-aware service introduction and automated provisioning) and that it is supported by our approach.

5 Validation and Results

Many applications that are currently being developed that require (or would benefit from) autonomic computing systems follow the premise of optimising the support for user-oriented services. Our framework is uniquely positioned to do this task efficiently and effectively, since it is *contextually aware* of changes to the environment and/or user. More important, it provides a means for reacting to context changes in a predictable and scalable manner through policy based management. Notably, this avoids requiring the use of skilled resources for simple, manually intensive operations. The automation that these kinds of architectures provide is critical for meeting the requirements of next generation services.

As an example application of this PBSM framework, we have been using a set of scenarios in which the main factor is the non-intervention of specialised network managers. Once the service has been created, the main idea is that the user interacts with the system in a direct way, and the network operator now has a much simpler and reduced role – to function mainly as a simple authoring entity, and/or an entity that allows the operation. Only occasionally will the network operator have to intervene and define new policies to take direct hard control of the system.

Let's imagine services that need to be deployed on the fly (e.g., services that are pre-defined but that will be deployed when the user signs in to the system or after a certain period of time). Such services can be functions that execute when the user logs in, a video conference or the downloading of multimedia content that starts on demand. Furthermore the services need to be assigned the appropriate quality of service, depending on the type of user as well as user-centric context-sensitive attributes (e.g., this is a user that has just purchased a new service, or this is an established gold user). These scenarios inherently require technological support based on policy that allows them to react in a specific manner based on the specific context.

Using Policies to manage a communication system requires more computer resources than systems that are not policy-based. Fig. 7 shows the CPU usage (%) and Memory usage (MBytes) versus the number of policies associated to a given service. We can observe that the CPU usage decreases with the replication of the service invocation rate. This means that the CPU usage decreases when the number of services (users of policy) increases, while the Memory usage increases due to the number of services (policies used to support services) being used. This occurs in all

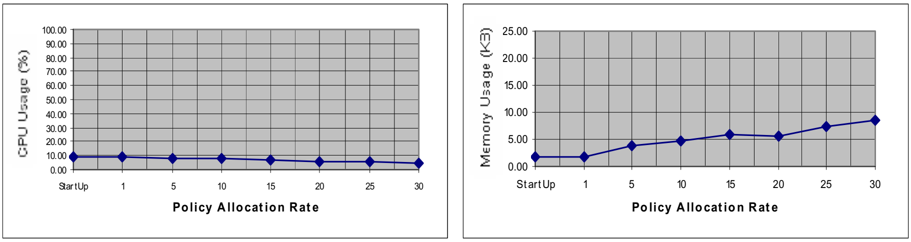


Fig. 7. CPU-Memory usage vs. Policy Allocation Rate

systems (even those without a policy-based paradigm). This feature is important when we try to design a scalable system with CPU limitations. In this scenario, the advantages in using a policy-based management system outweigh the benefit of requiring less computer resources, as provided by those that do not use policy.

The performance study was made using a PC with the following characteristics: Processor Intel Pentium 4 at 2.00 GHz. AT/At Compatible with 785 KB of RAM and supported by Microsoft Windows 2000 5.00.2195 with Service Pack 4; JRE 1.4.2_06 from Sun. and the AppPerfect DevSuite 5.0.1 - AppPerfect Java Profiler [20]

6 Conclusions

The main benefits from using policies for managing services are improved scalability and flexibility for the management of the systems; this also simplifies the management tasks that need to be performed. These scalability and simplification improvements are obtained by providing higher level abstractions to the administrators, and using policies to coordinate and automate tasks.

Policies make automation easier, as well as enable service management tasks to take into account any customisation of the service, made by either the consumer or the Service Provider. We have used the DEN-ng policy model due to its use of patterns, roles, and additional functionality not present in other models (e.g., its support of finite state machines). The extension of service management functionality to act *on demand* is an important property supported by using policy-based service management systems. We extend the current state-of-the-art in this area in two important ways. 1) We use the novel combination of policy management and context to describe changes to service management functionality and, more importantly, how to respond to these changes. 2) We have built a componentised, inherently scalable architecture for managing services. This architecture has then been linked into a larger autonomic architecture that uses the combination of information models, data models, ontologies, and machine-based learning and reasoning to achieve autonomic behaviour. Significantly, the Autonomic Manger of this architecture, and hence each of the components that it governs, are controlled by our Policy Manager.

The policy-based service management framework approach proposed emerges as a tool to solve some of the autonomic systems requirements in the line of the services management. This framework approach has been conceived to provide an innovative managing tool for next generation services and interact with autonomic systems.

Acknowledgments

This work is being extended and refers partially to the activities done in the framework of the IST-CONTEXT project, a two and a half years research and development project, partially funded by the European Commission. Thank you very much also to all the collaborators that helped in the realisation of this paper.

References

[1] Strassner, J. and Kephart, J., "Autonomic Networks and Systems: Theory and Practice", NOMS 2006 Tutorial, April 2006.

[2] Strassner, J., "Seamless Mobility – A Compelling Blend of Ubiquitous Computing and Autonomic Computing", in Dagstuhl Workshop on Autonomic Networking, January 2006.

[3] Kephart, J.O. and Chess, D.M., "The Vision of Autonomic Computing", IEEE Computer, January 2003. http://research.ibm.com/autonomic/research/papers/

[4] Strassner, J., "Policy Based Network Management", Morgan Kaufman, ISBN 1-55860-859-1

[5] G. Tomlinson, R. Chen, M. Hoffman, R. Penno, "A Model for Open Pluggable Edge Services", draft-tomlinson-opes-model-00.txt, http://www.ietf-opes.org

[6] Giacomo Piccinelli, Cesare Stefanelli, Michal Morciniec. "Policy-based Management for E-Services Delivery" HP-OVUA 2001.

[7] Westerinen, A.; Schnizlein, J.; Strassner, J. "Terminology for Policy-Based Management". IETF Request for Comments (RFC 3198). November 2001.

[8] Verma D. (2000) "Policy Based Networking" 1ª ed. New Riders. ISBN: 1-57870-226-7 Macmillan Technical Publishing USA.

[9] Moore, E.; Elleson, J. Strassner, A. "Policy Core Information Model-Version 1 Specification". IETF Request for comments (RFC 3060), February 2001.

[10] Moore, E.; "Policy Core Information Model-Extensions". IETF Request for comments (RFC 3460), January 2003.

[11] TMF, "The Shared Information and Data Model – Common Business Entity Definitions: Policy", GB922 Addendum 1-POL, July 2003.

[12] Damianou, N.; Bandara, A.; Sloman, M.; Lupu E. "A Survey of Policy Specification Approaches, http://www.doc.ic.ac.uk/~mss/MSSPubs.html

[13] ANDROID Project. http://www.cs.ucl.ac.uk/research/android

[14] Jun-Jang Jeng; Chang, H; Jen-Yao Chung; "A Policy Framework for Business Activity Management". E-Commerce, IEEE International Conference. June 2003.

[15] IST-TEQUILA Project. http://www.ist-tequila.org

[16] IST-CONTEXT project. http://context.upc.es

[17] Raz, and Y. Shavitt, "An Active Network Approach for Efficient Network Management", IWAN'99, July 1999, Berlin, Germany, LNCS 1653, pp. 220 –231.

[18] Henricksen. K., et al. "Modelling Context Information in Pervasive Computing System". Proceedings of Pervasive 2002, LNCS 2414, pp. 167-160, 2002.

[19] Please see http://www.omg.org/mda

[20] AppPerfect DevSuite 5.0.1-AppPerfect Java Profiler. http://www.appperfect.com/

Implicit Context-Sensitive Mobile Computing Using Semantic Policies

Hamid Harroud and Ahmed Karmouch

Multimedia & Mobile Agent Research Laboratory,
School of Information Technology & Engineering (SITE), University of Ottawa,
161 Louis Pasteur St. Ottawa, ON, Canada K1N 6N5, Canada
{hharroud, karmouch}@site.uottawa.ca
http://deneb.genie.uottawa.ca

Abstract. The availability of portable computing devices and advances in wireless networking technologies have contributed to the growing acceptance of mobile computing applications and opened the door for the possibility of seamless and pervasive services in mobile environments. However, with the dilemmas of limited device capabilities, network connectivity, transmission range and frequent changes, due to users and/or devices mobility, Internet-oriented wireless applications face challenges in terms of computation and interaction. These challenges elaborate with the inevitable demand for such applications to adapt to users' situations, their profiles, and the resources provided by the operating environment. Taking into account contextual information is an essential ingredient to cope with the frequent changes in mobile environments and hence provide adequate solutions for achieving adaptability, reliability, and seamless service provisioning. This paper introduces the use of policies over semantically modeled context as a technique for generating implicit context information and making use of context in mobile computing in general and internet-computing in specific. A context-sensitive instant messaging application demonstrates our proposed model using agent technology.

1 Introduction

Current dynamism in life is driving people's behaviour towards mobility and towards the usage of wireless devices. These devices are getting more powerful in terms of hardware characteristics such as power and memory as well as computing capabilities. Consequently, applications and services are developed to bring user's desktop environment to her/his hand-held devices. Some of these applications include emailing, chatting, web browsing, online banking, distributed games, multimedia streaming and recently web services [1]. In fact, services designed for use in mobile computing are expected to experience rapid growth. This transition from the desktop environment to the portable devices environment did not come free of problems but is overwhelmingly loaded with challenges such as content adaptation, mobility management, network reliability, and application interoperability to name but few.

Mobile computing needs solutions for providing dynamic interaction between users and applications on one hand and solutions for autonomic adaptation to users and environments context on the other hand.

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 188 – 200, 2006.

Context-aware computing has been advocated as a mechanism for providing such dynamic interactions and autonomic adaptation. Context awareness refers to the ability of an application or a service to adapt its behaviour in response to environmental changes. Accordingly, we designed a multi-agent system, MAS, that provides interoperability, adaptability and seamless service provisioning to users in mobile computing environment. MAS system is built on the concept of semantic modeling of context, its inference to policies, and the usage of inferred context policies enforcement.

Semantic can be defined as the study of meaning or change of meaning. When mapped to Internet applications it represents the ability of the applications to understand what they are performing as services, how to implicitly adapt to changes in the environment and how to interact with other entities.

Policies deal with the methods of manipulating applications and their profiles according to current situation. This includes where and how to store these profiles, how to automate adaptation, how to provide contextual access rights and how to retrieve relevant information at the right time, at the right place and to the right entity.

Our goal is to provide a model and infrastructure to support the development of context-sensitive applications and services by generating new contextual information using the semantic modeling of acquired context (*implicit context information*), and by inferring semantic context policies which, in turn, dynamically adapt the behaviour of different entities and services by enabling and executing appropriate policy actions (*implicit use of context*). In addition, the use of context level agreements (CLA) in the form of context negotiation [2] enables the ability to restrict an application agent's context to a subset of its specific policies. The CLA guides the interaction and information exchange among agents as they receive the agreed context in the form of semantic policies and make decisions accordingly on users' behalf.

The rest of the paper is organized as follows. Section 2 describes the multi-agent system in terms of agents' role and algorithms used. Section 3 discusses the automation process of manipulating contextual information and the semantic policies inference used by the system agents. Section 4 demonstrates an instant messaging application developed using the MAS. Section 5 presents the related work. And finally, section 6 concludes our work and highlights the future work.

2 Multi-agent System for Context Provisioning

We consider agents as the main computational entities of our system, as well as of applications and services requesting context information. Agents are provided with control abilities to govern their behavior and to make decisions autonomously by means of policies.

The introduction of policies inside agents increase their pro-activeness in facing new situations occurring in the surroundings due to users' mobility, devices' capability constraints, possibility of discontinuous interconnection and changes in the environment. The separation of the agents' coding logic (i.e. tasks) from their governing behaviors (i.e. policies) allows the reuse of agents' strategies and their overall behavior in order to dynamically adapt to various context information without significant impact on the mechanism used for their implementation.

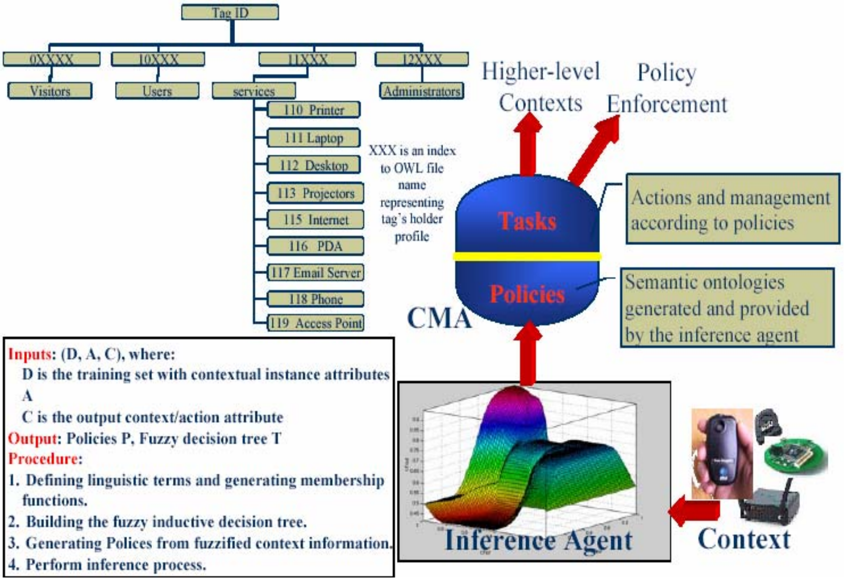
An agent becomes context-sensitive, meaning that it can sense changes in its context and adapt its behavior in response, by making use of semantic context policies.

Figure 1 depicts the proposed agents, context sources, presence interpretation and inference process. The proposed MAS system consists of two main agents: the context management agent and the reasoner agent.

2.1 The Context Management Agent

The context management agent (CMA) is responsible for monitoring on-going context-based sessions and managing the environment resources. The CMA has the authority to cancel, modify and/or renegotiate context specifications according to changes that might occur in the environment, or if agents violate the tasks assigned to them during negotiation [2]. Agents register with the CMA both to project their presence to other agents in the environment and to obtain the authorization policies needed to use available services and have access to other agents and information. The CMA stores relevant information in a knowledge base repository for inference, consistency maintenance and knowledge sharing. It models context and system information using ontologies implemented using the Ontology Web Language (OWL) [3].

It formulates context specifications from agents' requests and profiles according to the designed ontology constructs, axioms and rules of composition.



Inputs: (D, A, C), where:
D is the training set with contextual instance attributes A
C is the output context/action attribute
Output: Policies P, Fuzzy decision tree T
Procedure:
1. Defining linguistic terms and generating membership functions.
2. Building the fuzzy inductive decision tree.
3. Generating Polices from fuzzified context information.
4. Perform inference process.

Fig. 1. The multi-agent system model

2.2 The Reasoner Agent (RA)

The RA contains novel inference machinery that integrates logical reasoning, fuzzy reasoning and semantic rule representation in one system. Context captured from sensors and from users' and services' profile is treated as facts in the inference process. The reasoner agent uses these facts to deduce new context information and/or trigger actions. The reasoner agent uses logic-reasoning to insure that captured contexts are consistent both with each other and with constructs defined in the context ontology. It uses fuzzy logics to cope with the uncertainty in captured context and generates implicit context information which is then issued to different entities in the environment requesting context.

The MAS uses an RF-tag system for acquiring contextual information. We extended the RF-tags to provide other contexts than just identity and location. By taking advantage of the unique ID value of each tag, ontology profiles of entities are indexed in the system repository using these ID values. These ontology profiles include information such as activities of the user holding the tag, capabilities of the service with the assigned tag and the policy profile generated for each entity holding a tag in the environment. Figure 1 shows the categories we developed for the different entities in the environment and how they are associated to the tag IDs. For example a detected tag with the ID 111754 is automatically interpreted by the CMA to be a device of type laptop. From the detected tag it knows the laptop location while it uses the Tag ID number to fetch the OWL profile from the repository and thus gains knowledge of its implicit context and its governing policies.

The RA also uses these OWL profiles along with the inference algorithm to build an inductive fuzzy inference decision tree capable of extracting features embedded in the contextual information and inferring policies that will be used by the CMA for monitoring and governing the environment and entities interaction.

3 The Automation Process of Context Provisioning

As described in our work detailed in [4], we divided the process of automating context provisioning into three parts: generating implicit contextual information, inferring context to semantic policies governing the system's behaviour and supporting entities at runtime in accordance to the inferred policies. The first process requires a semantic modeling of context so that generating implicit contextual information could be preformed based on explicit context. Context inference is performed by transforming the ontology-based context instances to a set of linguistic terms over the context's range of values. Each term has a certain membership function to be used instead of the original context instance. As shown in figure 2, each entity in the environment (i.e. user, device, service …) is associated with two configuration files: the Agent Configuration File (ACF) and the Context Information File (CIF). ACF holds information about the agent that represents the entity (i.e. contact information for a user and the service description for a service with the reference to the CIF file location). These files are used by the inference agent to provide semantic policies and by the CMA to enforce and manage these policies.

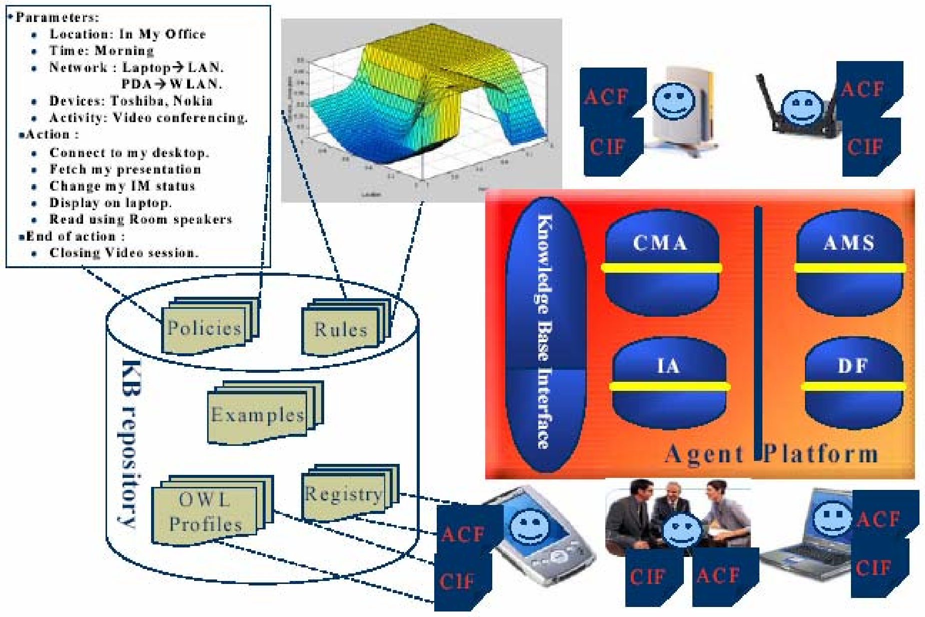


Fig. 2. Components used in the automation of context provisioning

A relation between a device and its owner is a sample example that illustrates the generation of implicit context. A crisp relation will have the following ontological representation.

```
<DeskTop rdf:ID="MMARL-10">
        <hasLocation rdf:resource="&Loc;B502"/>
        <belongsTo rdf:resource="& Confoaf ;Bob"/>
        ….
</DeskTop>
```

The captured context indicates that MMARL-10 is a device of type DeskTop, it is located in B502 and it belongs to user Bob. This information is acceptable if user Bob is actually the only person who is using this desktop all the time. But he might be using it only during the morning hours which will make this fact inaccurate and might cause unacceptable actions. When fuzzy logic is used in this situation to represent the uncertainty in the relation between the desktop and the user, the semantic representation will include the property hasSimilarity that will link the crisp context ontology to fuzzy ontology, then linguistic terms are defined that describe the membership functions used in the relation.

```
<Fuzz:hasFuzzyTerm >
 <Fuzz:FuzzyTerm rdf:ID="Morning">
  <Fuzz:hasMembershipFn>
   <Fuzz:trapezoidal>
     <Fuzz:hasFirstpnt>9.0</Fuzz:hasFirstpnt>
     <Fuzz:hasEndPoint>12.0</Fuzz:hasEndPoint>
     <Fuzz:hasScndpnt>11.0</Fuzz:hasScndpnt>
     <Fuzz:hasStartPoint>8.0</Fuzz:hasStartPoint>
   </Fuzz:trapezoidal>
 </Fuzz:hasMemebrshipFn>
 </Fuzz:FuzzyTerm>
```

```
<Fuzz:hasFuzzyTerm>
 <Fuzz:FuzzyTerm rdf:ID="Afternoon">
   <Fuzz:hasMembershipFn>
     <Fuzz:triangular>
       <Fuzz:hasCenter>13.0</Fuzz:hasCenter>
       <Fuzz:hasEndPoint>15.0</Fuzz:hasEndPoint>
     <Fuzz:hasStartPoint>11.0</Fuzz:hasStartPoint>
     </Fuzz:triangular>
   </Fuzz:hasMembershipFn>
 </Fuzz:FuzzyTerm>
</Fuzz:hasFuzzyTerm>
```

```
<Fuzz:hasFuzzyTerm>
    <Fuzz:FuzzyTerm rdf:ID="Night">
        <Fuzz:hasMembershipFn>
          <Fuzz:Crisp rdf:ID="Zero"/>
       </Fuzz:hasMembershipFn>
    </Fuzz:FuzzyTerm>
 </Fuzz:hasFuzzyTerm>
```

In this example, we defined three linguistic terms that describe the temporal relation between the desktop and the user. The first term is the “Morning” that defines the degree of certainty that Bob is using MMARL-10 during the morning from 8:00 AM till 12:00 PM. It is represented as a trapezoidal function which indicates that between 9:00 and 11:00 the system is certain that MMARL-10 belongs to user Bob. The second term is the “AfterNoon” which is described as a triangular function between 11:00 AM and 3:00 PM and the last term is the “Night” which is described as a crisp function with value equal to zero. This means that MMARL-10 is not belonging to Bob during the night.

With the aforementioned example that clarifies our intention of using fuzzy ontology to represent uncertainty in the context information, the next subsections describe the steps performed by the inference agent for reasoning about implicit context information.

3.1 Defining Linguistic Terms and Generating Membership Functions

Defining the linguistic terms and membership functions lie with our inference agent as users and application developers are assumed not to be expert in fuzzy logic and domain mapping techniques. Application developers will use the designed fuzzy-API to specify the number of linguistic terms to be used by the inference agent to map the context construct to them. For example, a developer may state that the location context inside a meeting room is specified by two linguistic terms “near” and “far”.

These two linguistic terms need to be mapped to the standard membership functions defined in the fuzzy ontology and stored in the system. For example, the two linguistic terms can be represented by Gaussian functions that have mean and variance determined by the range of the location and the overlap region. To generate the required membership functions, the context ontology is first used to check the range and the domain of the context construct and provides the result to the inference agent. The inference agent then uses a simple algorithm for membership function generation to achieve the low overhead in computation requirements. The algorithm starts by dividing the context range into 2n+2 equal slots, where n is the number of linguistic terms required (in the location example above, n=2). It then assigns four slots to each linguistic term; each adjacent terms share one slot in common that will represent the overlapping region in the fuzzy sets. The four slots concept is chosen because of the trapezoidal function that is considered to be the general membership function from which other membership functions can be derived. After successfully dividing the context range into its 2n+2 regions, the CMA randomly selects samples from each region and requests from the application developer, or user, to specify the expected membership function for each of these values. This step is required in order for the inference agent to correctly deduce the best membership functions that resemble user expectation. After this decision, the inference agent applies the Lagrange's interpolation mechanism on these samples data followed by Euclidian similarity measure to decide the standard membership functions that best resemble user-defined data.

3.2 Building the Fuzzy Inductive Decision Tree

After deciding the membership functions for the context information, the inference agent begins to build the fuzzy decision tree [5], shown at the bottom right in figure1, to perform the second process in the automation. Fuzzy decision trees improve the inference, robustness and generalization of classification, action triggering, and decision-making in context aware environments. In general, building an inductive decision tree requires a set of training examples, a heuristic method for choosing the best attribute to split at any node, and an inference process mechanism. Training examples in our system are acquired either from experts in the application domain or from sensors and historical actions of users. These examples are saved in the knowledge base and can be manipulated through the knowledge base API. The inference agent uses a heuristic splitting criterion that tries to predict the classification of context attributes. These values are used in conjunction with the entropy measurement to decide the winning attribute to use for splitting at any node.

The heuristic method used by the inference agent has two fold contributions. First, it reduces the example space through which the computation is performed, thus reducing complexity and increase speed of convergence. Second, it reduces the effect of a dominant linguistic term in the calculation because repeated values are counted using the fuzzy projection and then averaged. The root of the tree contains all the training examples. It represents the whole description space since no restrictions are imposed yet. Each node is recursively split by partitioning its examples. A node becomes a leaf when either its samples come from a unique class or when all attributes are used on the path. When it is decided to further split the node, one of the remaining attributes (i.e., not appearing on the current path) is selected. The examples present in the node being split are partitioned into child nodes according to their matches to linguistic terms of the wining attribute.

3.3 Policy Inference Process

From this inductive fuzzy tree, the IA proceeds to infer policies used in the inference process. Each path along the generated tree from root to leaf is converted into a policy with the antecedent part representing the context attributes of each succeeding branch, and the consequent part representing the class at the leaf with highest membership value. The membership value at each leaf is taken to be equal to the maximum membership value of all instances reaching that leaf and having that class as their output.

After generating these fuzzy policy sets from the tree, the inference agent sends them to the CMA in order to be used in the policy enforcement and provisioning of services. The process of inference with new instances and new implicit context information is processed from the generated tree. In this approach, context information is traversed along branches of the tree that satisfies its membership value. After reaching leafs, the class association value for each leaf is calculated. This is accomplished by taking the T-norm between the membership values of classes in the leaf, and the T-norm of that instance with the attributes along the branch from the root to the corresponding leaf.

3.4 Policy Specification and Enforcement

The last step in the automation of context provisioning is representing policies that govern the choices in behavior of the system in machine readable format.

We modeled policy information using OWL language [6] to easily integrate it with our developed ontologies for modeling context and fuzzy inference. The semantic policies in our MAS encapsulate the inferred policies from the fuzzy decision tree with meta-data information and enforcement role. The policy ontology has the following classes and properties:

- Policy (issuedBy*, appliesTo*, appliesWhen*, name*, appliesWhere*, priority*, equivalentTo*, composedOfPolicies*, policyRule*, enforcementType*, requiresOtherPolicies*)
  - Authorization ()
  - ContextSpecificPolicy (category*, description*)
  - Obligation ()
- PolicyEnforcement ()
    - instance Inform
    - instance Negative
    - instance Negotiable
    - instance Positive
    - instance Restricted
- PolicySimilar (degreeOfCertain*, similarIn*, similarTo*)
- Rule (hasAntecedent*, hasConsequent*, hasInferenceType*)

Policies are divided into: authorization and obligation policies [7]. We also have a generic policy class, ContextSpecificPolicy that can be used by application developers to specify its category and gives a description to it. Each policy has properties that state the entity issued the policy, the entities for which it applies the time and location of enforcing it. In addition the developer may state the equivalence of a policy with another one and state the degree of equivalence using ‘Similar’ class defined in the fuzzy ontology. This concept allows the MAS system to invoke other policies in case of uncertainty and vagueness and thus increases its robustness and performance. To allow the MAS system deduce what action to perform at a policy enforcement, every policy is associated with PolicyEnforcement value that could, for instance, have the value of ‘Negative’ to prohibit doing the rule in the policy, or it could be ‘Negotiable’ to allow agents to negotiate how to enforce this policy in case of conflict or inconsistency.

3.5 Semantic Context Modeling

The introduction of ontology, as a paradigm for modeling and representing the context, provides the ability to reason and deduce implicit information and enables the reuse of domain knowledge. As shown in figure 3a, we modeled the context in the form of expressiveness levels, starting from a high level of context abstraction down to levels, where context is expressed and refined in more detail. The highest level of abstraction is the ContextView which represents different types of context that belong to an entity (i.e. a user, a service, a device or the environment itself). ContextView has properties contains and invokes. The classes ContextFeatures and ContextEngagements are the respective ranges of those properties.

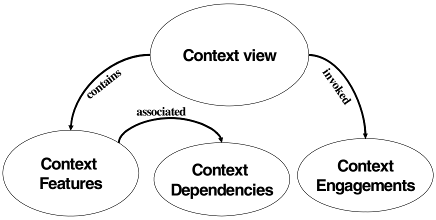


Fig. 3a. The upper context ontology

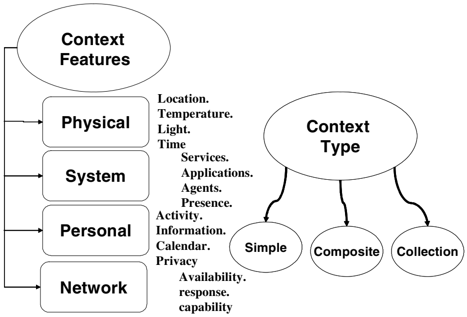
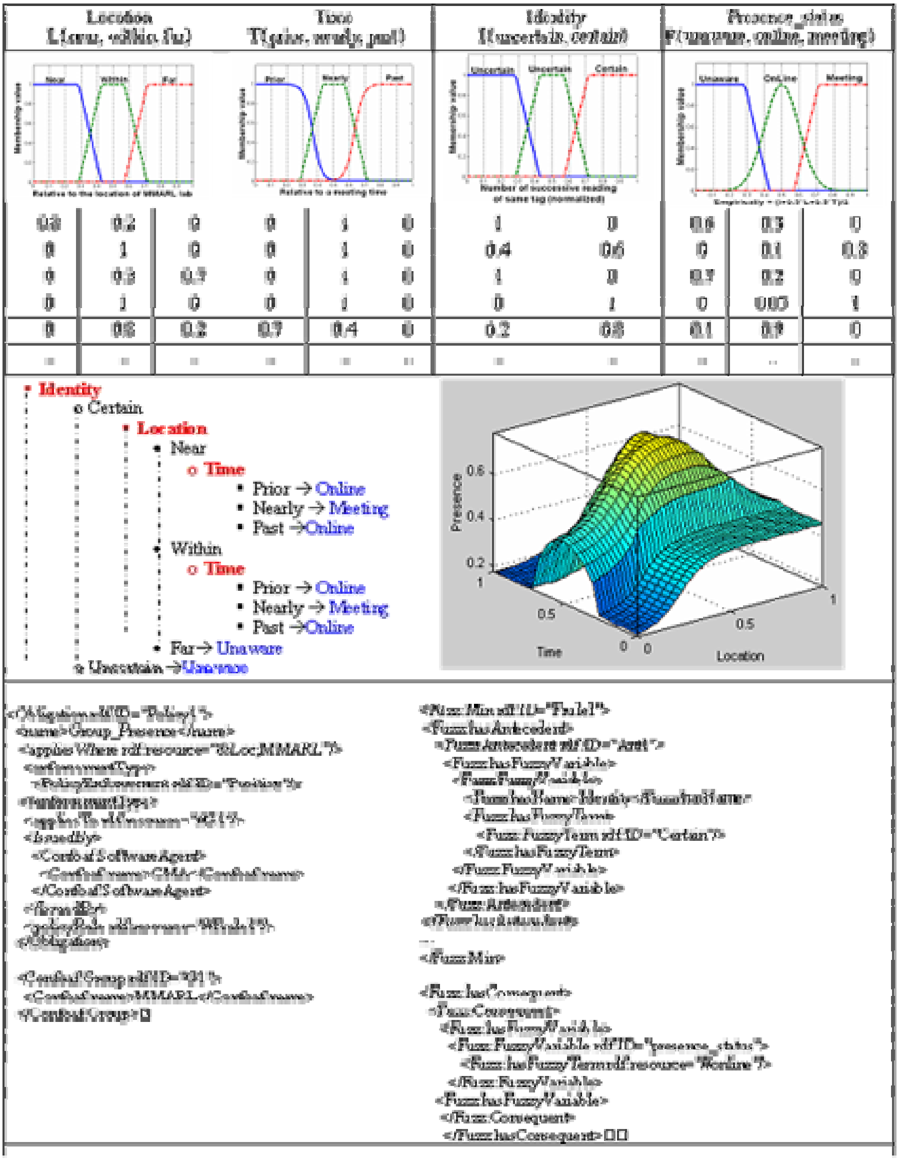


Fig. 3b. The defined context features and the associated context types

The ContextFeatures, shown in figure 3b, consists of classes related to the physical environment (i.e. location), classes related to the system (i.e. services), classes related to users (i.e. identities and activities) and classes related to the network (i.e. availability and bandwidth.) Each of these classes is categorized to be either simple, a composite of other context information, or a collection of similar context information [2].

4 Implementation

In order to test the feasibility of the proposed MAS system, we have implemented a context-aware instant messaging system (CIMS). CIMS is an extended version of buddy space open source [8] where we wrapped its interaction mechanism with agent technology and modified its internal structure to provide contextual information about users and services registered with our MAS system. CIMS provides tailored contextual information such as location, presence, event monitoring, activity alert …etc, based on the generated semantic context information and inferred context policies. ACF and CIF files for each entity are stored in the knowledge base repository. They are used by the inference agent to provide semantic policies and by the CMA to enforce and manage these policies.


Near
Within
Far
Membership value
Relative to the location of MMARL lab
Prior
Nearly
Past
Membership value
Relative to a meeting time
Uncertain
Uncertain
Certain
Membership value
Number of successive reading of same tag (normalized)
Unaware
OnLine
Meeting
Membership value
Presence
Time
Location
0.2
0.4
0.6
0
0.5
1
0
0.5
1

Location L(near, within, far)			Time T(prior, nearly, past)			Identity I(uncertain, certain)		Presence_status P(unaware, online, meeting)		
0.8	0.2	0	0	1	0	1	0	0.6	0.3	0
0	1	0	0	1	0	0.4	0.6	0	0.1	0.8
0	0.8	0.7	0	1	0	1	0	0.7	0.2	0
0	1	0	0	1	0	0	1	0	0.03	1
0	0.6	0.2	0.7	0.4	0	0.2	0.8	0.1	0.9	0
...	...	...	...	...	...	...	...	...	...	...

```
• Identity
      o Certain
            • Location
                • Near
                    o Time
                          • Prior → Online
                          • Nearly → Meeting
                          • Past →Online
                • Within
                    o Time
                          • Prior → Online
                          • Nearly → Meeting
                          • Past →Online
                • Far→ Unaware
      o Uncertain →Unaware
```

```
<Obligation rdf:ID="Policy1">
 <name>Group_Presence</name>
 <appliesWhere rdf:resource="&Loc;MMARL"/>
  <enforcementType>
   <PolicyEnforcement rdf:ID="Position"/>
 </enforcementType>
  <appliesTo rdf:resource="#G1"/>
  <basedBy>
   <ContextSoftwareAgent>
    <ContextName>CMA</ContextName>
   </ContextSoftwareAgent>
  </basedBy>
  <policyRule rdf:resource="#Rule1"/>
</Obligation>

<ContextGroup rdf:ID="G1">
 <ContextName>MMARL</ContextName>
</ContextGroup>□
```

```
<Fuzz:Min rdf:ID="Rule1">
<Fuzz:hasAntecedent>
  <Fuzz:Antecedent rdf:ID="Ant1">
   <Fuzz:hasFuzzyVariable>
    <Fuzz:FuzzyVariable>
     <Fuzz:hasName>Identity</Fuzz:hasName>
     <Fuzz:hasFuzzyTerm>
      <Fuzz:FuzzyTerm rdf:ID="Certain"/>
     </Fuzz:hasFuzzyTerm>
    </Fuzz:FuzzyVariable>
   </Fuzz:hasFuzzyVariable>
  </Fuzz:Antecedent>
</Fuzz:hasAntecedent>
...
</Fuzz:Min>

<Fuzz:hasConsequent>
 <Fuzz:Consequent>
  <Fuzz:hasFuzzyVariable>
   <Fuzz:FuzzyVariable rdf:ID="presence_status">
    <Fuzz:hasFuzzyTerm rdf:resource="#online"/>
   </Fuzz:FuzzyVariable>
  <Fuzz:hasFuzzyVariable>
  </Fuzz:Consequent>
  </Fuzz:hasConsequent>□□
```

Fuzzification of the semantic context is provided by the application developer and stored in the knowledge repository and is used to generate the initial training examples that will be used to build the inductive fuzzy tree. The life cycle of context provisioning is depicted in the table above.

The first row represents the context information types used by CIMS to deduce the presence of entities in the environment. The second row shows the membership functions used to fuzzify these three contexts.

We divided the location information of the fifth floor in Colonel-by building at the university of Ottawa with respect to the MMARL laboratory into three categories (near, within and far) while the IA chosen the respective membership functions of these three categories according to the algorithm in section 3-1, the same process applies to the other two contexts as shown in the second row. Rows from 3 till 8 illustrate samples of the training examples used by the inference agent to build the inductive fuzzy tree (IFT). The inference agent builds the IFT shown in the first column of the 9th row while the second column depicts the rule surface extracted from the IFT. Sample of the inferred policies are shown in the last row of the table, where the first column represents a positive obligation policy of the rule shown in the second column of the last row, {*If Identity is certain, Location is near, Time is prior then Presence is unaware}*. The policy is applied to MMARL group, issued by the CMA and applies only to MMARL lab.

Figure 4a shows a snapshot of the CIMS user interface of user Bob. The figure shows the services that can be used by this user, the HP printer and the MMARL projector, according to his location, time and generated policies. The figure also shows that the projector is currently busy, as it has the red light, and also shows that his friend Balo is in a meeting. Hamid is unclear to user Bob which is an indication that the MAS system is un-aware of the presence status of Hamid. It means that Hamid could be present in the environment but his presence is vague.

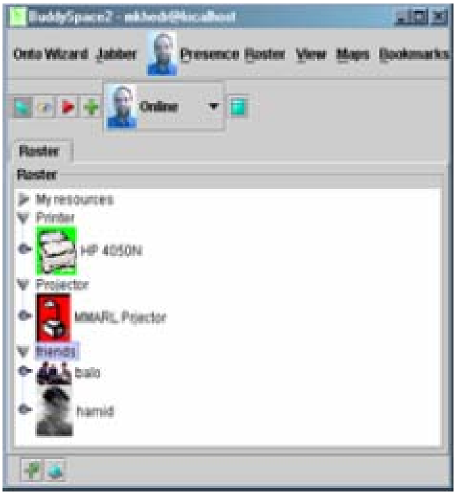


Fig. 4a. Snapshot of the CIMS user Interface

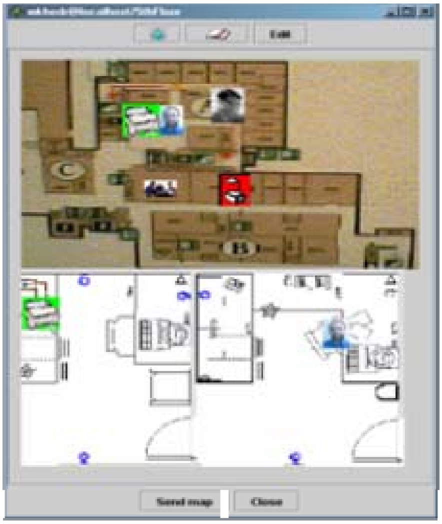


Fig. 4b. The map GUI with entities activities and location projected on it

Figure 4b is a map that projects the activity and location of entities found in the user's presence list. It is a two layer map where the above layer is the map of the fifth floor while the lower map is the map of Bob's office. Every user is projected in the map so that Bob can have knowledge about their location context spontaneously as they move in the fifth floor. Finally, every user can manipulate his ACF and CIF through the OntoWizard menu item and at the same time browse and search profiles of other entities if their policies authorize such actions.

5 Related Work

While the field of context-aware computing is relatively young, a number of powerful context-aware systems such as Context Toolkit [9] and Hewlett Packard's Cooltown [10] have been developed, but most of them shared a weakness in supporting knowledge sharing and context reasoning because of their lack of ontology [11]. This weakness has been addressed by projects like GAIA [12], PSI [13] and VTT research [14]. GAIA brought the functionality of an operating system to physical spaces. Common operating system functions are supported, such as events, signals, file system, security, processes, process groups, etc. Gaia extends typical operating system concepts to include context, location awareness and mobile computing devices.

PSI described a novel support for attaching context-aware services to physical locations or objects which enables context-aware interaction by utilizing context information from the user, network, and sensors; as well as automated use of available wireless and mobile infrastructure relative to what the application tries to accomplish so as to minimize exposing such decisions to the user.

VTT project proposed a new software framework that simplifies the development of context-aware mobile applications by managing raw context information gained from multiple sources and enabling higher-level context abstractions.

While these projects have successfully built systems that effectively interact with users and the environments' context, they do not incorporate the usage of inductive inference to generate new contextual information from the acquired ones (implicit context) and to automate the process of context provisioning by its inference to semantic policies (implicit usage of context). In addition exchanging inferred context policies and enforcing them using agent technology is a novel technique.

6 Conclusion and Future Work

In this paper, we presented a novel approach for supporting the context-awareness of applications and services in mobile computing. Our implicit context-sensitive technique for inferring new contextual information, a set of semantic policies, and distributing these policies among a number of agents provides various applications and services to be context-aware. As the context changes, new contextual information is generated and semantic context policies are dynamically created to reflect the changes in the state of the environment. The creation of semantic policies provides spontaneous adaptability to applications and services for handling vagueness in context information. Automated context provisioning through policies also helps developers to rapidly build more reliable context-aware applications and services.

In addition, the separation of system behaviour, in the form of policies, from the system implementation is also a key factor in our MAS system that enables interoperability and system reusability (i.e. encapsulating context in policies eliminates the need for rewriting context management code across applications).

We plan to develop more applications to validate different aspects of the MAS system. This includes inter-domain context provisioning, dynamic tuning to the inductive process of policy generation and context negotiation among multiple domains.

Acknowledgment

This work was partly supported by a Research Grant from the University of Pierre et Marie Curie, ParisVI (France), and the Natural Sciences and Engineering Research Council of Canada.

References

[1] N. Davies, H. Gellersen, “Beyond Prototypes: Challenges in Deploying Ubiquitous Systems” IEEE Pervasive Computing, pp. 26-35, Vol.1, No.1, January-March 2002.

[2] M. Khedr, A. Karmouch, " A Semantic Approach for Negotiating Context Information in Context Aware Systems” IEEE Intelligent Systems Magazine, 2005.

[3] McGuinness,D.L.; “Question answering on the semantic web” Intelligent Systems, IEEE Volume 19, Issue 1, Jan.-Feb. 2004.

[4] M. Khedr, A. Karmouch, “An Infrastructure for Managing Context Information in Pervasive Computing Environments,” PhD Thesis, University of Ottawa, SITE, 2004.

[5] S.R Safavian, et. al, “A survey of decision tree classifier methodology”, IEEE Trasn. Systems Man Cybernet Volume21, Issue 3, May-June 1991.

[6] “OWL Web Language Ontology Reference,” http://www.w3.org/TR/owl-ref.

[7] A.K. Bandara, E. C. Lupu, A. Russo, “Using Event Calculus to Formalise Policy Specification and Analysis”, 4th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2003), Lake Como, Italy, June 2003.

[8] http://www.buddyspace.org

[9] D. Salber, A. Dey, G. Abowd, “The Context Toolkit: Aiding the Development of Context-enabled Applications”, CHI (1999) 434–441.

[10] T. Kindberg, J. Barton, “A Web-based Nomadic Computing System”, Computer Networks 35(4) (2001) 443–456.

[11] H. Chen, T. Finin, J. Anupam, “Semantic Web in a Pervasive Context-Aware Architecture”, Proceedings of Artificial Intelligence in Mobile System 2003.

[12] M. Român, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt “Gaia: A Middleware Infrastructure to Enable Active Spaces.”, IEEE Pervasive Computing, Volume 1, Issue 4, pp. 74-83, Oct-Dec 2002

[13] T. Kanter, "Attaching Context-Aware Services to Moving Locations", IEEE Internet Computing, Volume 7, Issue 2, March-April 2003, pp.43 - 51.

[14] Panu Korpipää, el al, “Managing Context Information in Mobile Devices”, IEEE Pervasive, pp. 42-51, Vol.2, No.3, July-September 2003.

GXLA a Language for the Specification of Service Level Agreements

Badis Tebbani and Issam Aib

Laboratoire Informatique de Paris 6 (LIP6)
Université de Paris 6
8 rue du capitaine Scott
75015 Paris France
(name.surname)@lip6.fr

Abstract. In this work we propose GXLA, a language for the specification of Service Level Agreements (SLA). GXLA represents the implementation of the Generalized Service Level Agreement (GSLA) information model we proposed in a previous work. It supports multi-party service relationships through a role-based mechanism. It is intended to catch up the complex nature of service interactivity in the broader range of SLA modeling of all sorts of IT business relationships. GXLA is defined as an XML schema which provides a common ground between the entities in order to automate the configuration. GXLA can be used by service providers, service customers, and third parties in order to configure their respective IT systems. Each party can use its own independent SLA interpretation and deployment technique to enforce the role it has to play in the contract. An illustrative VoIP service negotiation shows how GXLA is used for automating the process of SLA negotiation and deployment.

Keywords: *Self Configuration; Service Level Agreement; Policy Based Management; SLA Specification, Policy Specification.*

1 Introduction

Policy-Based Network Management (PBNM) is concerned with the use of rules to manage the configuration and the behavior of one or several entities in an IT infrastructure. A rule connects a set of conditions with a set of actions. When the conditions are verified, the corresponding actions are activated. The evaluation of the conditions can be triggered by some events.

In the context of service-oriented management, service level agreements (SLAs) are a very useful tool. Their necessity is continuously increasing, essentially for resource optimization and configuration automation. In the context of service provider networks, where collaboration and services exchange takes precedence, Policy-Based Management is necessary to react quickly to the changing environments without human intervention. There is currently no existing standard for specification of SLAs. Accordingly, there are many proposals that are more or less adapted to the information technologies (IT) context. The specification must be

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 201–214, 2006.

generic in order to be applicable to all possible scenarios and extensible to be able to add specific parameters.

Note that, in the literature, we can distinguish three domains of studies that overlap and are complementary. The success of a PBNM depends on these three solutions: (i) The representation and exchanges of the hardware configurations [1,2], where the stress is laid on the performance and the scalability of the PBM system, generally, it is supposed that the database is already filled by policy rules with respect to the SLAs. (ii) The second field of study is policy refinement, it concern representation, checking, validation, detection/resolution of the conflicts and the enforcement of the policies. This field overlaps with the first one especially in hardware configuration level. Besides it also proposes tools needed for the storage of policies [3,4]. (iii) The third field of study is about modeling the external world of the system, like the business environment, the service negotiation and the technical parameters describing the service [5,6]. In this field, a precise representation is needed in order to make services understood by all parties in both business and hardware level.

In this work, we focused on the last field of studies: modelling and formal specification of the service level agreements that make able to clearly describe the customer requirements and the objectives of the company. The majority of the existing specifications in the literature are specific to the Web services, and do not allow to describe several providers in the same service. We propose a language XML-based, role-oriented and multi-parties, who allows specifying several parties involved in the service (provider, subcontractors...) by associating each one to one role that it have played in the agreement. Our specification offers a simple interface to the users (or to the networks administrators) to request a service (or to introduce policies) in a business language, then it translates this contract into corresponding configuration policies.

The paper is organized as follows. In the next section, we will examine related work on the specification of SLAs. Existing work on the modeling of the policies will be presented in the third section. Our GSLA information model and its specification into the GXLA XML schema will be explained in the fourth section. The next section will show an example of execution to test our specification and we will conclude by some perspectives for our work.

2 Work on the Specification of Service Level Agreements

2.1 WSLA Web Service Level Agreement IBM2003

WSLA [5] is a good case of study where the steps and the course of the design of a language of specification of the SLA are explained. WSLA is XML-based. In WSLA (Fig. 1), the SLA class contains three parts: the first describes the parties involved in the service, the second contains one or more descriptions of the service and the third describes engagements. It defines signatory parties to be either a service provider or service customer. They are the main parties of the SLA and are assumed to "sign" the SLA, bearing the final liability. Additional supporting parties can also exist and are sponsored by one or both of the signatory parties.

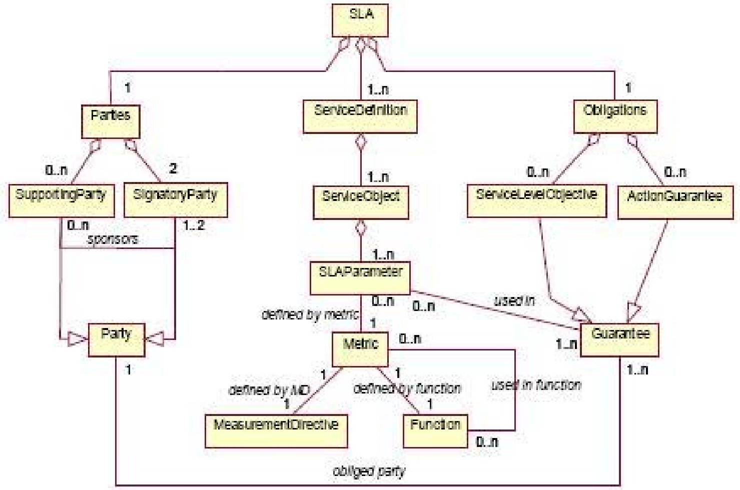


Fig. 1. Overview of main WSLA components [5]

The description of the service contains one or more operations which define one or more parameters. In their turn the Parameters are connected to metrics. A metric can be: (i)Connected to a directive which indicates how it must be measured. (ii) Connected to a function, in this case the metric one can be the combination of one or more other metric. Finally the Engagement class contains two classes of obligations: no or several SLO which guarantee a state of the SLA for one determined period, the other type of engagements is the actions promised in the SLA at the time of particular events.

The WSLA insists on the process of measurement of metric and articulates its specification to allow to the service provider (or customer) to sub-contract this process with a third company. WSLA is based on the descriptors of services Web WSDL, SOAP, UDDI which facilitates the description of the service and its indexing.

2.2 SLAng

SLAng is a SLA language, under development as an element of the project TAPAS in United College, London [6]. It is a XML-based language and which rests on the description languages of the Web services (WSDL) and the server applications (J2EE, CORBA).

SLAng defines seven types of SLAs. Initially, it divides the SLA into two main categories: (1) Horizontal SLA: the contract between two equal entities (of the same level of architecture), eg. Two applications. (2) Vertical SLA: the contract between two entities in different layers, eg. The networks layer with the application. SLAng conceives the SLA like a contract between strictly a customer and a provider without taking into account the other actors in the network. SLAng introduces the concept of client responsibility, provider responsibility and the mutual responsibility (eg. penalties). Each one of it describes the obligations of each part. Our GXLA extends this concept in the role and defines the responsibility or the guarantees for a service independently, to be able to then assign it to one or more parties.

3 Work on the Specification of Policies

3.1 PONDER Imperial College, London

The Ponder(Imperial College, London 1992-2005) offers a framework to specify access rights and management policies with an role-based access control [3,7]. Ponder is a declarative, object-oriented language, based XML and supports the typing and the combination of the policies. Ponder consider a policy as the choice of setting parameters of a network component, or the authorization access according under certain conditions. It defines several types of policies witch articulated on ECA-paradigm (Event, Condition, and Action).

Ponder defines four types of policies: (1) *Authorizations:* are essentially security policies related to access-control and specify what activities a subject is permitted or forbidden to do, to a set of target objects. They are designed to protect target objects so are interpreted by access control agents or the run-time systems at the target system. (2) *Obligations:* specify what activities a subject must do to a set of target objects and define the duties of the policy subject. Obligation policies are triggered by events and are normally interpreted by a manager agent at the subject. (3)*Refrains:* specify what a subject must refrain from doing and are similar to negative authorisation policies but are interpreted by the subject. (4) *Delegations:* specify which actions subjects are allowed to delegate to others. A delegation policy thus specifies an authorisation to delegate. These four types are the bases of all the policies, Ponder defines also four other types of composition of policies to group the policies belonging to the same field to simplify the specification: *groups*, *roles*, *relationships* and *management structure*. Ponder also defines the limits of the applicability of the policy and it introduces the concept of *Domains* to group the policies which concern, the same subject or the same geographical area.

```
inst auth+ VideoConf1 {
     subject  /Agroup + NYgroup;
     target    USAStaff-NYgroup;
     action    VideoConf(BW, Priority);
     when      Time.between(1600, 1800)
        and(Priority > 2);
        }
```

Fig. 2. Ponder policy Example

In Ponder, the actions are achievable files, may be of the external scripts stored in precise fields. The access to these fields is controlled by the *Authorization* policies. Lastly, the Ponder specification is a very good candidate for the implementation of the SLAs. But, it must raise its level of abstraction to take into account the applicative needs and all the structure of the SLAs. Work on Ponder has been also stopped for two years and we do not see well how it will evolve.

Figure 2 show an example of a Ponder policy : The members of Agroup and Bgroup can accept carry out a videoconference with the personnel based in the United States with the group of New York. The constraint of the policy is made up here. The time constraint limits the service to apply only between 4:00pm and 6:00pm and the constraint of action indicates that the policy is valid only if the priority parameter is larger than 2.

3.2 WS-POLICY (IBM, Microsoft and al. 2003)

WS-Policy [8] is a framework which defines the basic structures for the description and the communication of the policies of the Web services. It forms a flexible and extensible grammar to express the needs, capacities and preferences of a service Web in a format XML. The policies are defined in WS-Policy like a collection of "alternatives". The alternative is a collection of assertions. The assertion is the basic unit of the policies. Each assertion belongs to a specific field. WS-Policy defines the applicability of the following policies: *Security*, *Privacy*, *Priorities application*... etc. The assertions can be gathered or combined by the operators <*wsp:All*>, <*wsp:ExactlyOne*> and <*Optional*>.

WS-Policy does not define the way in which the policies will be interfaced with the Web services; it leaves the free field to other specifications that define the way in which the policies are attached to the Web services for a better adaptability. For example, *WS-PolicyAttachement* is a specification which defines the association of WS-Policy with WSDL and UDDI. The principal idea of WS-Policy is to offer a grammar to describe the policies (preferences, capacities...) of each Web service and then to be able to exchange this information and to include/understand their semantics. WS-Policy is integrated in messages SOAP, this one being extensible to support new types of messages. Lastly, WS-Policy

```
<wsp:Policy>
   <wsp:ExactlyOne>
    <wsse:SecurityToken>
     <wsse:TokenType>wsse:
      Kerberosv5TGT</wsse:TokenType>
     </wsse:SecurityToken>
     <wsse:SecurityToken>
     <wsse:TokenType>wsse:
     X509v3</wsse:TokenType>
    </wsse:SecurityToken>
   </wsp:ExactlyOne>
</wsp:Policy>
```

Fig. 3. WS-Policy Example

defines three operations for the treatment of the policies: *Normalize*, *Merge* and *Intersect*. These operations define a model to compile the policies by standardizing each policy initially and then to combine them according to a some rules.

Figure 3 represent two specific security policy assertions that indicate that two types of authentication are supported. A valid interpretation of the policy above would be that an invocation of a Web service contains one of the security token assertions specified.

4 The GSLA Framework

4.1 GSLA Information Model

A GSLA [9] is defined as a contract signed between two or more parties relating to a service relationship and that is designed to create a clear measurable common understanding of the role each party plays in the GSLA. A party role represents a set of rules which define the minimal service level expectations, and service level obligations that the party has with other roles and at which constraints. The constraints might be of any type and normally include contract scope (temporal, geographical, etc.), the agreed upon billing policies, as well as the expected behavior in case of abnormal service operation.

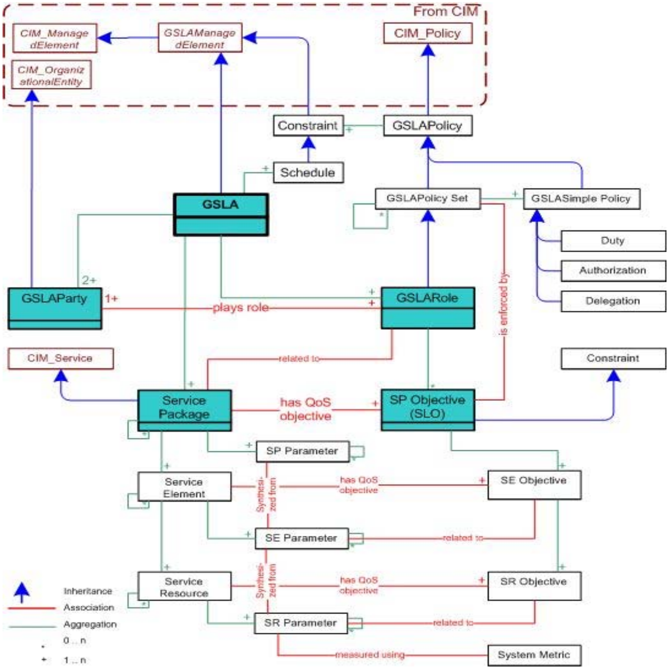


Fig. 4. GSLA [9]

We identify in Fig. 4 the top-level components of the GSLA. A GSLA comprises a set of parties joined according to a certain schedule in order to realize the contract by playing each one or more roles. During the GSLA life cycle, a required behavior or constraint related to the GSLA is captured in the model by the abstract GSLAPolicy component. A *GSLARole* is modeled at first approximation by a set of GSLA rules, and as it participates in defining the behavior of the system, it is derived from the *GSLAPolicy* component. A Schedule component represents the temporal scope during which a GSLA component is valid. Finally, a GSLA object comprises one or more *Service Packages* to each of which is associated a *Service Package Objective* that some GSLA party is required to guarantee as is specified in its attached role(s).

A *service package* is an abstraction that enables different service elements (customer facing services) to be packaged together. Each *Service Element* is related to one or more *Service Resources*(SR). A service element is enabled through a set of service resources. A *service resource* is intended to be transparent to the customer and represents a basic provider resource, such as an email server, a network element, a processing server, a database, a shared file, or a stockpile.

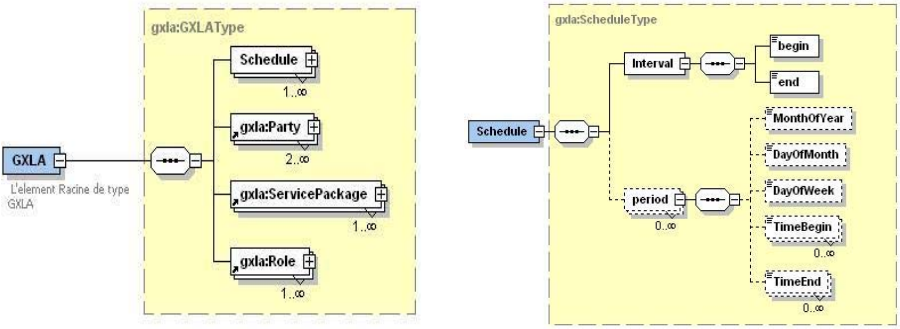
We propose a way for modeling QoS that caters for the specification of both high-level business objectives as well as that of low-level resource QoS parameters. The *Service Package Objective* (SPO) component defines Service Level Objectives for one or more SPgs. Basically, an SPO is a constraint and we allow it to be defined in two possible ways: First, as a set of predicates or logical expressions over one or more SPg Parameters. This represents a high-level way of defining QoS objectives based on direct calculus made over high-level service parameters that are synthesized from basic System Metrics up through System Resource (SR) parameters and System Element (SE) Parameters. The other way around is to calculate the objectives based on QoS appreciations coming from subordinate *Service Element Objectives.* This represents the high-level compilation of low-level QoS appreciations. This latter approach reflects better the way users appreciate a given service infrastructure; i.e. by giving a final appreciation based on separate 'sub' appreciations over the different service components. In the GSLA information model, multiple party service relationships are supported and each party has a set of SLOs to assure and some behavior to follow with respect to the other parties. Also, to each SLO are generally associated rules (or policies) that define actions to take in case the SLO has not been respected or some threshold has been reached. We structure related SLOs and their corresponding policies into the Role class. Roles can be of several types: compulsory (such as supporting a specific routing scheme in an ad hoc network), optional (SuperNode in a p2p file sharing community), required by at least one party (Central Controller in an IBSS), or statically attached to a specific party (a VoD Provider) at the GSLA specification time.

A party behavior is captured within the *GSLARole* component. As Roles are ultimately translated into low level policies, a party behavior at the lowest view is modeled as a policy. A policy is of two types, either a duty or an authorization. A duty defines conditions that need to be met in order to execute some system operations. They represent the key to QoS policy specification in our model. A GSLARole is modeled through the set of duty and authorization policies having their subject domain the party or the group of parties that play that role; as well as the set of SLOs that it is required to ensure as part of its responsibilities in the SLA.

4.2 GXLA, an XML Specification for the GSLA

The GXLA is a XML-Schema which implements the GSLA information model. It is usable at the same time by the service provider and the customer of the service in order to configure their respective systems. Moreover, it allows the internal use by the provider for compilation and the configuration of its system in order to provide the concluded services. The SLA in the GXLA is composed of four sections (see Fig. 5.a):

1. *Schedule*: represent the total temporal range of the contract. It is structured like a set of time intervals delimited by a beginning and an end (Fig. 5.b).



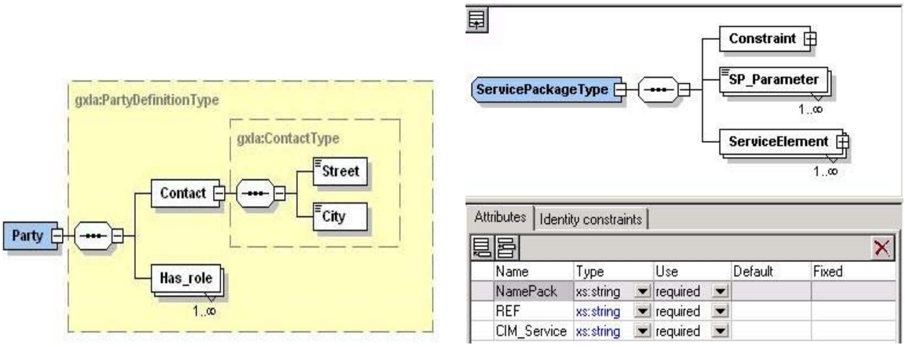
(a) Main GXLA components

(b) Schedule

Fig. 5. Global GXLA(a) and GXLA.Schedule(b)

2. *GXLAParty*: represents the parties involved in the service. Unlike to the preceding specifications, the GXLA describes general information of each party without specifying its nature (provider or customer) in order to cover all the possible states in the services networks market. The element *Has_role* (Fig. 6.a) permit to attach each party to one ore several roles that it will play in the service.
3. *ServicePackage*: is an abstraction to describe one or more services (Fig. 6.b). It is composed of:
   - *Constraint*: specify the temporal range of each service within the limit of the temporal range of the including components, in fact the *Schedule* constraint of the SLA.

- *SP-Parameter*: is the formalization of the high level parameters and represents the common sight of the service by all the participants. We define them in the GXLA like results of synthesis of metrics (*SE-Parameter*). *SP-Parameter* or *SE-Parameter* can be attached to one or more SLO which supervise QoS of the service.
- *ServiceElement*: is an abstraction to define the service in question, its operations, its constraints and its parameters. The GXLA makes it possible to describe the service with high level parameters (vocabulary of the contract) visible at the customers. Then, it attaches each service to one or more resources system (*ServiceResource*) in a transparent way to the customers. Indeed, each parameter of the service (*SE-Parameter*) is the result of a function or a directive on the metric ones of the system (eg. SNMP report). The metric can be atomic (eg. measure on a resource) or made up of several other metrics (eg. average of the number of server invocations).



Name	Type	Use	Default	Fixed
NamePack	xs:string	required		
REF	xs:string	required		
CIM_Service	xs:string	required		

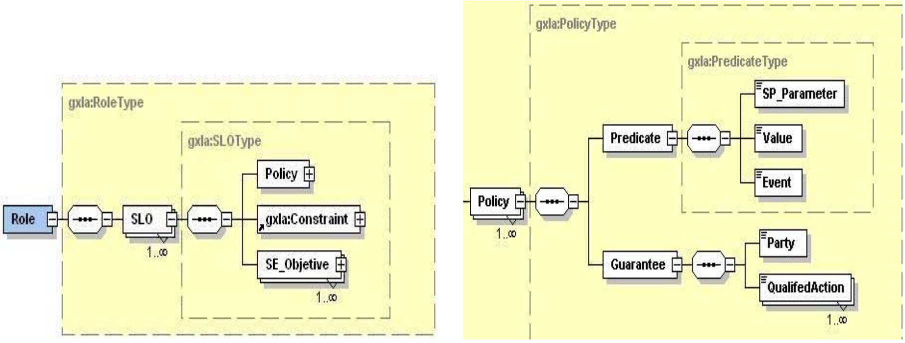
(a) Parties

(b) Service Package

Fig. 6. GXLA.Parties(a) and GXLA.ServicePackage(b)

4. *Role*: For each service, QoS corresponds guaranteed by the provider and waited by the customer. The GXLA described in the *Role* element, the SLOs which will define the level of service necessary for each service (Fig. 7.a). A SLO, in the GXLA, is ensured by one or more policies (Fig. 7.b). The policies model in the GXLA are event-oriented, the *predicate* checking is done each time when event arrive; the *SP-Paramter* is compared to the *Value*. If the condition is verify, *QualifedAction* is triggered.

   The policies have many types: (i) policies of obligation which define the *QualifedAction* and the party responsible for the role under certain conditions on *SP-Parameter* (eg. execution of a schell, notification in the case of a violation or a value close to a threshold). Obligation policies are the key of the QoS guarantees.(ii) policies of authorization which define the access rights for certain actions on a system resource under certain conditions, in particular on the metric ones (eg. to change the size of a queue).



(a) Role

(b) Policy

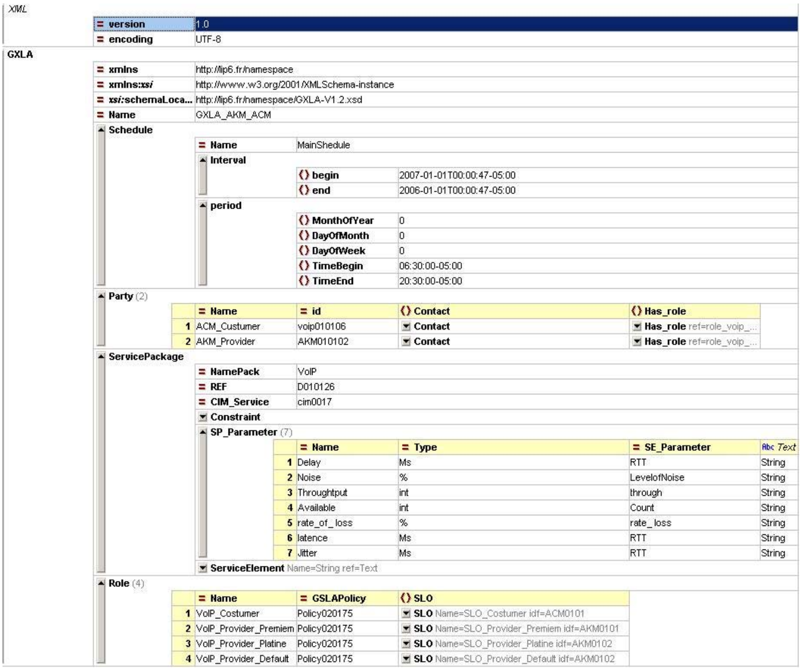
Fig. 7. GXLA.Role(a) and GXLA.Policiy(b)

5 Use Case: VoIP Service

In this section, we consider a scenario where a customer requests a Voice over IP (VoIP) service. The context of our scenario includes a service provider owning a network with a geographic coverage and an underlying PBM system and it offers service through a Web site. Through the web site portal, the customer chooses his services and after signing the contract he starts to use them. The implementation comprises two parts: the customer part which is an Applet and the provider part which is a set of Servlets that intercept information coming from the customer and treat them.

We test our specification through two aspects: service negotiation and automatic generation of suitable policies. The GXLA must allow collecting personal user information on one hand and the characteristics of the service on the other hand. In addition, starting from information on the level of quality of service desired by the customer, the GXLA is used along with policy generation templates to infer the suitable policies. For the creation of service level agreements, we prepared a generic contract GXLA (see Fig. 8) which comprises all the available services with all existing levels of available qualities of service . After the negotiation, the servlet creates an instance of the generic contract (GXLA) and personalizes it with: customer information, subcontractors (if they exist) and especially suitable policies. Tables 1 and 2 show some typical policies for the VoIP service with two levels of QoS.

We considered in our implementation that the provider proposes its various services through a Web site. The customer thus has a convivial interface which posts the services that a customer can request without any skills in computer networks. This interface (Fig. 9), which conforms the GXLA, permits the customer to introduce his personal information, desired service(s) and level of quality of service. In a step forward, the provider receives the customer request under the shape of a XML file, respecting GXLA specification, which is parsed in order to extract necessary information. Then, using XSL, the provider produces a HTML



XML

- version: 1.0
- encoding: UTF-8

GXLA

- xmlns: http://lip6.fr/namespace
- xmlns:*xsi*: http://www.w3.org/2001/XMLSchema-instance
- *xsi*:schemaLoca...: http://lip6.fr/namespace/GXLA-V1.2.xsd
- Name: GXLA_AKM_ACM
- Schedule
  - Name: MainShedule
  - Interval
    - begin: 2007-01-01T00:00:47-05:00
    - end: 2006-01-01T00:00:47-05:00
  - period
    - MonthOfYear: 0
    - DayOfMonth: 0
    - DayOfWeek: 0
    - TimeBegin: 06:30:00-05:00
    - TimeEnd: 20:30:00-05:00

Party (2)

	Name	id	Contact	Has_role
1	ACM_Custumer	voip010106	Contact	Has_role ref=role_voip_...
2	AKM_Provider	AKM010102	Contact	Has_role ref=role_voip_...

ServicePackage

- NamePack: VoIP
- REF: D010126
- CIM_Service: cim0017
- Constraint
- SP_Parameter (7)

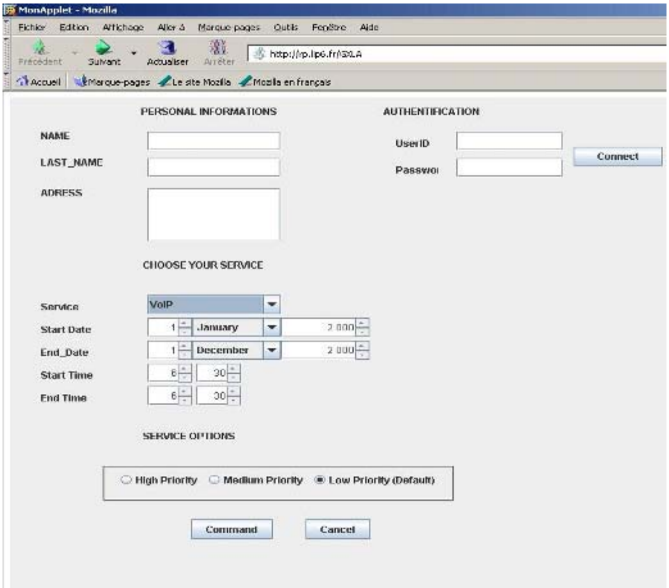
	Name	Type	SE_Parameter	*Text*
1	Delay	Ms	RTT	String
2	Noise	%	LevelofNoise	String
3	Throughtput	int	through	String
4	Available	int	Count	String
5	rate_of_ loss	%	rate_ loss	String
6	latence	Ms	RTT	String
7	Jitter	Ms	RTT	String

- ServiceElement Name=String ref=Text

Role (4)

	Name	GSLAPolicy	SLO
1	VoIP_Costumer	Policy020175	SLO Name=SLO_Costumer idf=ACM0101
2	VoIP_Provider_Premiem	Policy020175	SLO Name=SLO_Provider_Premiem idf=AKM0101
3	VoIP_Provider_Platine	Policy020175	SLO Name=SLO_Provider_Platine idf=AKM0102
4	VoIP_Provider_Default	Policy020175	SLO Name=SLO_Provider_Default idf=AKM0102

Fig. 8. A Global Specification of a VoIP with three levels of quality of service



MonApplet - Mozilla
Fichier Edition Affichage Aller à Marque-pages Outils Fenêtre Aide
Précédent Suivant Actualiser Arrêter
http://rp.lip6.fr/GXLA
Accueil Marque-pages Le site Mozilla Mozilla en français
Connect
Command
Cancel

PERSONAL INFORMATIONS

NAME

LAST_NAME

ADRESS

AUTHENTIFICATION

UserID

Passwo

CHOOSE YOUR SERVICE

Service: VoIP

Start Date: 1 January 2 000

End_Date: 1 December 2 000

Start Time: 6 30

End Time: 6 30

SERVICE OPTIONS

○ High Priority ○ Medium Priority ● Low Priority (Default)

Fig. 9. User interface

version of the contract including details like the service(s) price which is send back to the customer permitting him to choose whether to accept or refuse the agreement. If he accepts, the contract is agreed then the provider proceeds to turn up the service and notify the customer.

Through this example, we showed the role of GXLA in the process of automatic services negotiation. Once the contract is concluded, the provider must parameterize its equipment in an optimal way in order to honour the contract. Next paragraph is dedicated to show the role of GXLA in such a procedure and especially in the translation of the SLA agreement into service level objectives. The provider translates the agreed contract according to GXLA specifications:

1. He will initially define the role of each participant: itself and the customer.
2. The provider by defining the role of each one lays down the policies which must be applied in order to guarantee the contract. The policies in our example are deduced from to rules illustrated in table 1 and 2.

Table 1. Example of the policies for the VoIP-Premiem

	Predicate	Unit	Action	Etat
P1	Delay $>$ 99	μs	Notification	Active
P2	Jitter $>$ 9	μs	Re-connect + Notification	Active
P3	Rate-of- loss $>$ 0,98 then	%	Gold + Notification	Active
P4	Noise $>$ 5	%	Penalties = Penalties+1 %	Active

Table 2. Example of the policies for the VoIP-Platine

	Predicate	Unit	Action	Etat
P1	Delay $>$ 199	μs	Notification	Active
P2	Jitter $>$ 19	μs	Re-connect + Notification	Active
P3	Rate-of- loss $>$ 1,98	%	Gold + Notification	Active
P4	Noise $>$ 10	%	Penalties = Penalties+1 %	Active

- P1: is a policy which takes care the *time of transmission* parameter, we check if it does not exceed certain thresholds. If the value of the parameter exceed, an alarm is trigger to the provider.
- P2: is a policy that observes *Jitter parameter.* If it exceeds certain thresholds the provider prefers to notify and to reconnect the customer.
- P3: is a policy which takes care the *rate of loss* parameter. If it exceeds certain thresholds, the provider rocks the flow of the customer in the priority flow.
- P4: is a policy that observes the *rate of noise* parameter, if it exceeds certain thresholds, the sum of the consequently penalties augment.

Once created, the contract is stored in a XML-database. An agent is then dedicated to derive the policies from each contract and stores them in the policies database (Policy repository). Then, the PEP (Edge-Router) controls the user access by requesting the PDP (central controller) which responds back with the policies to be applied for the corresponding user. Once policies are stored in the database, the steps of consistency checking and conflict detection are to be done, independently of the technical enforcement policies.

During the use case implementation, we note:

- GXLA is sufficient to describe the services provider environment and the case of the request/negotiation of services. The provider has at the end of the negotiation a concise specification of the contract, with a specification of the suitable policies.
- The fact that we chose XML like language of specification enabled us to benefit from many tools available for the treatment: XML-SPY or Eclipse to edit and validate our specification, XSLT style sheets to convert our XML files into other formats like posting the contract to the customer; JAVA especially the Document Object Model (DOM) library for the parsing and the instantiation of XML files.
- However, future work is essential to define the semantics of the policies. Right now, the policies semantic (parameters, values and actions) are defined informally, witch has turned out to be weakness. It is necessary to define for each service a number of parameters with the corresponding thresholds.

6 Conclusion

In this work, we presented GXLA, an XML-based language which implements the GSLA information model for the specification of the service level agreements (SLA). GXLA is role-oriented, each role includes a set of SLOs and rules which characterize each party's behavior in the SLA. We illustrated the usability of the GXLA through a VoIP service negotiation along with the generation of the full contract script and its enforcing policies. The specification of the GXLA language represents the first step in the automation of service-oriented management. Ongoing work considers the refinement of the policy generation process and the actual configuration of the IT platform to enable the contracted SLAs for real applications.

References

1. Boutaba R, and Al.: Extending COPS-PR with Meta-policies for Scalable Management of IP Networks. (2002) International Journal on Networks and Systems Management (special issue on Management of Converged Networks, vol.10, No.1, pp.91-106.
2. Kamel, H., al.: Designing Scalable on Demand Policy-based Resource Allocation in IP Networks. (IEEE Communications Magazine)
3. Sloman, M., al: The Ponder Policy Specification Language. Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, 29-31 Jan.2001, Springer-Verlag LNCS 1995, pp. 18-39 (2001)

4. Moses T, and Al.: extensible access control markup langage(xacml). (Feb. 2003) Technical report, OASIS, version 1.0.
5. Ludwig, H., Keller, A., Dan, A., P.King, R., Franck, R.: Web Service Level Agreement (WSLA)
Language Specification. IBM Corporation (2003)
6. D.Davide Lamanna, J.S., Emmerich, W.: SLAng : A Langage for Defining Service Level Agreements. (2003) IEEE FTDS.
7. Dulay, N., Damianou, N., Lupu, E., Sloman, M.: A policy language for the management of distributed agents. In: AOSE. (2001) 84–100
8. Bajaj, S., Al.: Web Services Policy Framework (WSPolicy). (IEEE Consumer Communications and Networking Conference) IBM, Microsoft, and al. september 2004.
9. Issam Aib, and al.: The Generalized Service Level Agreement Model and its Application to the SLA Driven Management of Wireless Environments. (ACIT 2004)
10. Aib, I., Agoulmine, N., Pujolle, G.: A Multi-Party Approach to SLA Modeling Application to WLANs. IEEE Consumer Communications and Networking Conference (Jan 2005)
11. Aib, I., al.: Capturing Adaptive B2B Service Relationships Management throug a Generalized SLA Information Model. HPOVUA (2004)
12. Heiko Ludwig, Alexander Keller, A.D.R.K.: A Service Level Agreement Language for Dynamic Electronic Services. (IEEE Network 2002)
13. Verma, D.C.: Simplifing network administration using policy-based management. IEEE Network (2002)
14. Sahai, A.: Automated sla monitoring for web services (2002)
15. Machiraju, V., Sahai, A., van Moorsel, A.: Web service management network: An overlay network for federated service management (2002)
16. TeleManagementFORUM: SLA Management Handbook GB917. (juin 2001)
17. Ganz, A., Ganz, Z., Wongthavarawat, K.: Multimedia Wireless Networks: Technologies, Standards and QoS. Prentice Hall PTR (2004)
18. Distributed Management Task Force, I.: Common information model (cim), policy model white paper (18 June 2003) CIM Version 2.7.
19. Network Working Group, R..: Policy core information model (February 2001) Version 1 Specification.
20. IETF NETCONF Working Group: (http://www.ietf.org/html.charters/netconf-charter.html)

A Service Management Approach for Self-healing Wireless Sensor Networks

Helen P. Assunção[*], Linnyer B. Ruiz, and Antônio A. Loureiro

Electrical Engineering Department
Federal University of Minas Gerais
Av. Antônio Carlos, 6627
31279-010, Belo Horizonte, MG, Brazil
{helen, linnyer}@cpdee.ufmg.br, loureiro@dcc.ufmg.br

Abstract. Wireless Sensor Networks (WSNs) can provide three types of basic services: sensing, processing and disseminating. A shortcoming of any of these services can disturb the network goals. However, failures are not exceptions in WSNs, since problems as energy exhaustion, communication range loss or physical damages are usual incidents. Considering the need of managing WSNs in an efficient manner, improving information quality and availability, this work proposes the use of the IT Infrastructure Library (ITIL) and the autonomic computing paradigm in the design of self-healing WSNs. We also propose a service management approach for a self-managed network that dynamically adapts itself in order to maintain the service availability and promote the resources productivity. This approach aims to employ the self-healing service in WSNs, allowing them to discover, examine, diagnose and react to dysfunctions. Results show that service management applied to a self-healing WSN extend the longevity and availability of the network.

1 Introduction

A Wireless Sensor Network (WSN) is a distributed sensing tool and in this work, viewed as an Information Technology (IT) system. In the majority of cases, these networks are composed of hundreds of thousands of elements (sensor nodes), which are able to collect, process, disseminate and store data. The elements perceive the environment, monitor different parameters and collect data according to the application purpose.

The design of autonomic systems for WSNs must consider that this kind of network has particular characteristics that distinguish them from traditional networks, such as severe communication, energy and processing constraints, and deployment in remote and inhospitable environments without human intervention. For this reason, any hardware or software operation must be energy efficient, including self-management operations.

Due to the need of managing WSNs in an efficient manner, improving information quality and availability, moreover reducing costs, this work proposes

[*] The author was supported by the National Council for Scientific and Technological Development (CNPq).

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 215–228, 2006.

the use of the IT Infrastructure Library (ITIL) and the autonomic computing paradigm in the design of an self-healing WSNs.

Autonomic computing [3] is a technology that defines systems that manage themselves and improve their operation without direct human intervention. ITIL [7] is a set of best practices to manage IT infrastructure and is the most widely accepted approach to IT service management in the world. Together these technologies incorporate an accepted model and independent of underlying structure, technology or architecture to the autonomic computing, meanwhile applies the ITIL processes to system with low level of human intervention.

As a study case, this work employs some of the ITIL concepts to provide the self-healing service, in case of WSN nodes failure. The system design involves events detection, notification and classification. Whenever this system detects improper operations or sensor nodes failures, the autonomic managers negotiate resources in an effort to recover the network automatically, keeping the services availability. The detected events can lead the system to conflict situations, which imply the need of learning skills to solve them. The proposed system works as a combination of strategies and policies that allows the description of the system behavior, the failures context and other events. These are information of major concern since they improve learning and planning tasks in a WSN.

This work is organized as follows. Section 2 presents Wireless Sensor Networks (WSNs) that self-manage themselves without direct human intervention. This section also presents the main management services for an autonomic WSN. Section 3 makes a brief presentation of the service support and service delivery processes of the Information Technology Infrastructure Library (ITIL). Section 4 presents one of the important contributions of this work - the definition of an Autonomic Service Management System, developed based on the autonomic computing and ITIL concepts. Section 5 presents a study case, in which the model proposed is instantiated for a WSN, considering the self-healing service. This section also presents the performed experiments as well as the obtained results. Section 7 aims to show some of the related work autonomic WSNs. Concluding the work, section 8 presents the final comments and future works.

2 Autonomic WSNs

WSNs assign a set of technological resources to the generation and employment of information. The network produces, processes and delivers its own data. The network is the user of its own services and can negotiate sensing, processing, storing and delivering services, according to the established goals.

In the majority of WSNs applications, network elements are deployed in remote areas where the maintenance and administration by technicians are impracticable. These elements, called sensor nodes, are designed with small dimensions and are expected to cost few dollars allowing the use of hundreds of nodes in different applications. Sensor nodes are composed by a computational unity, a wireless communication unity, a sensing unity (one or more sensors), a logic unity (software) and a power unity, which recharging is generally impracticable (in face of the great amount of nodes and the deployment in remote areas).

Sensor nodes perceive the environment, monitor different parameters and collect data according to the application purpose. In certain types of application, the network must collect, process and deliver data in validity period. In other types of applications, sensed data must be delivered to an application that correlates them and generates a report on the environment. Any missing or late information can influence the correlation results. For this reason, a failure in the element sensing, disseminating, storing and disseminating activities can disturb the network goals.

To design and develop energy efficient management systems in environments that impose severe restrictions is not a trivial task. Considering WSN characteristics, this work proposes that they should be autonomic, that is, they should self-manage themselves with the least human intervention.

An autonomic system is composed by interrelated autonomic elements. Each of these elements has managed resources, that is hardware or software that build the IT infrastructure, and autonomic managers that supervise and control these resources using an standard interface (touchpoint). The autonomic manager provides self-management services using monitoring, planning, analyzing and executing modules. Figure 1 presents the interaction between autonomic elements [2].

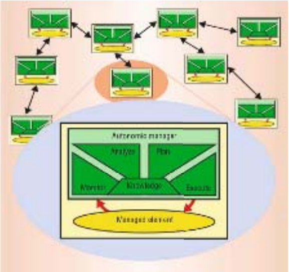


Fig. 1. Autonomic system

Regarding to autonomic WSNs, the management tasks considers one or more situations perceived from the environment. These tasks should consider some aspects [11]:

- *Self-healing:* the management service that discovers, diagnoses and reacts to network disruptions. Self-healing components detect improper operations and failures and start corrective actions based in defined policies to recover the network or a node. The automatic recovering from damages improves the service availability.

- *Self-optimization:* the management service that maximizes the resource allocation and utilization, and guarantees optimal service quality, based on policies. The automation of complex tasks and the components adjustment in response to variable workloads allows the delivery of a high-level service.

- *Self-configuration:* the management service that changes configuration parameters to adapt itself dynamically under varying conditions and network states. This service self-configures and reconfigures the network elements under varying and even unpredictable conditions. The network configuration must occur automatically, as well as dynamic adjusts to the current configuration to best handle changes in the environment.
- *Self-protection:* the management service that detects and protects the network against threats (internal or external, accidental or malicious). In case an attack happens, this service executes detection routines in order to reach security.
- *Self-service:* the management service that allows the provision of sensing, processing and dissemination services, anticipating resources and at the same time keeping the complexity hidden, in order to shrink the gap between business application and service goals.
- *Self-awareness:* the management service that allows the entity to know its environment and its activities context and act accordingly. It finds and generates rules to best interact with neighbors entities.
- *Self-knowledge:* the management service that qualifies an entity that knows itself. For example, an entity that governs itself should know its components, current state, capacity and all the connections with other entities. It needs to know the extension of its resources that can be lent and borrowed.
- *Self-maintain:* the management service that allows an entity to monitor its components and fine-tune itself to achieve pre-determined goals of an entity. In

this work, the autonomic WSN is composed of autonomic sensor nodes - the smallest part of the autonomic system. Four common functions, identified in the characteristics describe above, should be implemented in the autonomic sensor nodes: a function to collect the details it needs from the system; a function to analyze those details to determine if something needs to change; a function to create a plan, or sequence of actions, that specifies the necessary changes; and a function to perform those actions. These functions work together to provide the control loop functionality of an autonomic manager [2]:

Monitor. The monitor function encloses system (hardware, software) and environment monitoring in order to extract the behavior of these components. This function collects details from the managed resources, and aggregates, correlates and filters them into symptoms that can be analyzed. The details can include topology information, metrics, configuration, status, capacity and throughput.
Analyze. The analyze function provides mechanisms to observe and analyze situations in an effort to determine whether changes need to be implemented. This function can model complex behaviors to use prediction techniques allowing the autonomic managers to learn about the IT environment and predict future behaviors.
Plan. The plan function creates or selects a procedure to execute a desired change in the managed resource. A change plan, which represents a desired set of changes for the managed resource, is created and passed to the execute function.

Execute. The execute function provides the mechanism to schedule and perform the necessary changes to the system. This function is responsible to execute the change plan and to update the knowledge used by the autonomic manager.

This work uses some of the IT Infrastructure Library (ITIL) concepts to employ self-management services in autonomic WSNs. For this purpose the monitor, analyze, plan and execute functions are implemented for autonomic managers defined under the ITIL paradigm.

3 The IT Infrastructure Library

The IT Infrastructure Library (ITIL) has become the most widely accepted approach to IT service management. It provides a consistent set of best practices for IT service management, promoting a quality approach to achieving business effectiveness and efficiency in the use of information systems [1].

The IT Infrastructure Library, documented in approximately forty books, describes the main processes of IT service management. The two main areas of ITIL are the Service Support and Service Delivery. Together, these two areas consist of ten disciplines that are responsible for the provision and management of effective IT services.

The Service Support disciplines include the Incident Management, Problem Management, Changes Management, Release Management and Configuration Management [5]. The Configuration Management discipline provides information about the IT infrastructure and controls this infrastructure by monitoring and maintaining information about all the necessary resources to deliver services. The Incident Management recovers the service functioning as fast as possible. In addition to that, it minimizes the incident impact on business operations and guarantees the offer of the best quality of service and availability, according to the Service Level Agreement (SLA). The Problem Management stabilizes the IT services by minimizing incidents consequences by the removal of their root cause, preventing incidents, problems and recurring incidents. The Change Management coordinates and plans changes in an efficient manner, with the minimum risk to the existing infrastructure. The Release Management implements the proposed changes, guaranteeing the use of authorized, tested and correct software and hardware while implementing these changes.

The Service Delivery disciplines include the Availability Management, Continuity Management, Capacity Management, Financial Management and Service Level Management [6]. The Availability Management predicts, plans and manages the service availability, considering aspects of reliability, maintainability and redundancy. This goal is reached by determining business availability requirements and adjusting these requirements to the capacity of the IT infrastructure. The Continuity Management plans alternative configuration items in an effort to recover from problems. The Capacity Management identifies and specifies the client's demands, and then, translates these needs into resources, guaranteeing the services performance. The Financial Management identifies, computes and manages the cost of delivering IT services. The Service Level Management maintains and improves the quality of IT services, by monitoring and adjusting these services.

These disciplines interact to guarantee that the IT infrastructure delivers to the business a high quality service. In the next section, ITIL is presented as an integration solution to employ self-management services and functions (monitor, analyze, plan and execute) in autonomic WSNs.

4 Service Management System for Self Healing for WSNs

Considering that failures are not exceptions in WSNs, this work deals with the design of a self-management system for WSNs that detects and identifies failures, and proposes adjusts to the network infrastructure, in order to maintain the service availability.

The architecture of the proposed system is described as follows. Based on Service Level Agreements (SLA), the system generates service management policies to the network elements. These policies are delivered to the network nodes and stored in repositories, the knowledge bases. The autonomic managers, located in the sensor nodes, start the monitor, analyze, plan and execute functions, based on these information.

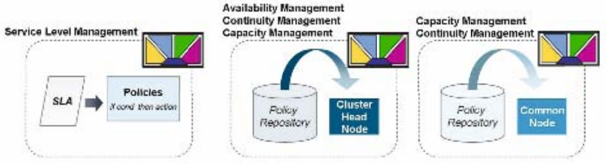


Fig. 2. The elements of an autonomic service management

Each autonomic manager supervises some managed resources through the monitoring function. Their analyze function searches for policies stored in the knowledge base to determine if changes need to be implemented due to the occurrence of incidents. The plan function adjusts an existent solution to solve the problem according to its extension. Once a change plan is defined, the execute function applies it to the system and updates the knowledge source.

The proposed service self-management system consider the utilization of hierarchical WSNs, that is, networks in which sensor nodes are organized in clusters and each cluster has at least one leader. Autonomic managers and a policy repository, the knowledge base, are instantiated in the logic component of the sensor nodes, according to the nodes role in the network - common node or cluster head node role.

An autonomic manager, located outside the network, is responsible to map the Service Level Agreement (SLA) into policies for the network nodes, to monitor the service quality and availability, and if necessary, to renegotiate the SLA.

Autonomic managers that control cluster head nodes are responsible to guarantee that the service level is being attended inside the cluster and, otherwise, to adjust the network components to attend these levels.

Common nodes autonomic managers are responsible to monitor their resources, optimize the nodes functioning, detect anomalous behavior, analyze events and adjust the nodes configuration in order to diminish problem risks. In case any problem occur, these managers will recover the network functioning as fast as possible. Common nodes include in their logic component not only autonomic managers and the knowledge source, but also touchpoints to interact with managed resources.

Some concepts of the ITIL Service Delivery area were employed in the definition of four autonomic managers with the purpose of creating a self-healing WSN, namely:

- Autonomic Service Level Manager: the autonomic manager that guarantees the fulfillment of agreed service levels and, eventually, redefines the SLA, using a manual manager or policies.
- Availability Autonomic Manager: the autonomic manager that plans and manages service availability through the monitoring of the IT service availability.
- Continuity Autonomic Manager: the autonomic manager that analyzes network risks identifying possible failures and creating a recover or risk reduction plan.
- Capacity Autonomic Manager: the autonomic manager that monitors nodes resources and identifies demands. In case of current or future insufficient capacity this manager is responsible to reallocate resources and anticipate new resources, what makes necessary the definition of a resource utilization model to determine whether the nodes are attending the defined requirements or not.

Each one of these managers employs concepts of Service Support disciplines to accomplish the monitor, analyze, plan and execute function, considering the self-healing service (see Figure 3).

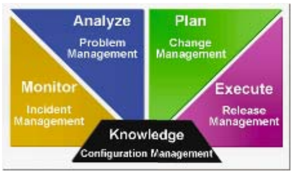


Fig. 3. Autonomic element

The monitor function will detect events that are not part of the service normal operation and that can cause a disruption of the service or diminish its quality. The analysis function will perform the incident analysis, error diagnosis and root cause determination, and then determine if changes need to be performed triggering a Request for Change (RfC) if necessary. The plan function will propose changes to sensor nodes in order to solve network problems. The execute function will implement the proposed change plan in the network elements, and evaluate the change impact over the network.

Information utilized and generated by these four functions are stored in the nodes knowledge sources. The WSN autonomic managers are responsible to generate, manage and store information about the sensor nodes resources in the knowledge source.

The managers defined in this section will contribute to maintain the WSNs service availability.

5 Study Case

This section presents a study case utilizing the service management approach, proposed in this work, for a hierarchical WSN, in which incidents were instantiated as communication problems. The system must detect the root cause of the problems - in this study case problems are caused by traffic congestion and energy loss. After detecting improper operations or components problems the system will automatically recover from disruptions, maintaining service availability and continuity. Autonomic managers use residual energy data and production information as source of information to the monitor and analyze functions.

The knowledge bases are distributed between the network nodes. Common nodes store information about their own state. Cluster head nodes store not only these information but also network management information, once these nodes have more communication and storage capacity.

The main tasks defined to the autonomic managers located in common nodes are described bellow.

Monitor. The continuity manager detects message loss incidents. In order to detect if messages were lost, the nodes periodically check if they received an ACK message for each sensed data message sent. The capacity manager is responsible to monitor the sensor node residual energy. When the energy decays a determined threshold, an incident is detected and the node starts disseminating only high priority messages. Besides, this manager is able to detect abnormal production increase incidents, which occurs when some event of interest is sensed making nodes to increase their production.

Analyze. The error diagnose is accomplished using information stored in the sensor nodes knowledge sources. In case common nodes lose messages, the autonomic manager will diagnose if an abnormal increase in nodes production is the root cause of the incident (this increase makes nodes lose their messages since the packet queue is full).

Plan. After the autonomic manager detects that messages were lost and the production has increased, it will try to decrease the node's production, until messages are no longer lost. In order to control their production the manager proposes in the changes plan gradual increase of the sensing and dissemination interval. In case of a low energy problem the manager will propose the stop the node activities for some seconds.

Execute. The manager alters the sensing and disseminating intervals or puts the node out of service for a few seconds. While messages are being lost, the manager keeps adjusting these parameters and evaluating their impact over the network.

The main tasks performed by each one of the autonomic managers located in the WSN cluster head nodes are described bellow.

Monitor. The detection of message loss is accomplished by the continuity manager using the same ACK mechanism of the common nodes. The capacity manager is responsible to monitor the sensor node residual energy. Cluster head node managers, also detect the network increase of production incident and low energy level in the cluster.
Analyze. In case cluster head nodes lose messages, these nodes analyze the knowledge source in order to diagnose if the failure is an abnormal production increase.
Plan. If the autonomic manager detected a low residual energy incident in its cluster, it proposes as change that the 20% of the cluster nodes with the smallest residual energy will stay out of service for a 5 seconds interval. A similar change is performed when the network has an abnormal production increase (which is characterized by a significant raise in the number bytes sent). The cluster head node chooses 20% of the cluster nodes with the highest production and put these nodes out of service for a 5 seconds interval. These plans aim to increase the network lifetime and deliver data rate.
Execute. The autonomic manager execute the change plan putting the chosen nodes out of service for a few time, and then evaluates the impact of the change over the network.

The knowledge source is updated whenever messages are received or configuration parameters are changed. Local information stored in the repository are: *id*, *energy*, *bytesSent*, *sens_interval*, *diss_interval*, *drop_number*, *last_drop_number*, *bytes_sent*, *bytes_dropped*, *bytes_dropped_second*, *bytes_second*, *last_bytes_second* and *priority*.

The cluster head nodes store in the knowledge source their local information and some replied information of each node of the cluster, namely: *id*, *energy*, textitdiss_interval and *bytes_sent*.

Each message received by the cluster head from an unknown node triggers the action of creating an entry for this node in the knowledge source (KS). After that, this repository will be updated by messages sent periodically from the common nodes with information about their repository entries. Besides, the cluster head knowledge source entries are modified when a sensed data message received, updating the received message rate from each node of the cluster.

The management application described in this section was simulated and the experiments are presented in the next section.

6 Experiments

This section presents the simulation scenarios and experiments simulated using the Network Simulator 2 (NS-2) tool and the MannaSim [15], which is a framework made of a set of base classes that extends NS-2 to simulate sensor networks. The simulation scenarios were built with dimensions of 50 x 50m containing: four cluster head nodes and five common nodes per cluster. The cluster head nodes were positioned in a grid and the common nodes deployed randomly. An access point (AP) was positioned in the center of the scenario. Using this topology, two scenarios were simulated.

Scenario 1: The service management system for self-healing WSNs described in section5 was implemented.

Scenario 2: the network does not implements self-management functionalities that is, autonomic computing services were not implemented.

Table 1. Characterization of performed simulations

Network Configuration	**Simulations Configuration**
Cluster Head Nodes Number: 4;	*Simulation Time:* 155 seconds;
Common Nodes Number: 20;	*Number of Simulations:* 33;
Transport Protocol: UDP;	*Scenario Size:* 50 x 50m;
MAC Protocol: IEEE 802.11;	
Cluster Head Nodes Configuration	**Common Nodes Configuration**
Range: 250 metros;	*Range:* 40 metros;
Processing Consumption: 0.360W;	*Processing Consumption:* 0.024W;
Transmission Consumption: 0.6W;	*Transmission Consumption:* 0.036W;
Reception Consumption: 0.3W;	*Reception Consumption:* 0.024W;
Sensing Consumption: -	*Sensing Consumption:* 0.015W;
Disseminate Type: Programmed;	*Disseminate Type:* Programmed;
Sensing Type: -	*Sensing Type:* Programmed;
Battery Capacity: 100J;	*Battery Capacity:* 5J;
Bandwidth: 100kbps;	*Bandwidth:* 28.8kbps;

The simulations characterization is presented in Table 1. The specifications of the two kinds of sensor nodes utilized in the simulations, common and leaders, where configured according to the ones presented by the real nodes Mica Motes [4] and WINS [14], respectively. Each scenario was executed 33 times and the results, presented in section 6.1, refers to the average of the obtained values.

The implemented application accomplishes the temperature monitoring using sensor nodes thermistors. The average sensed temperature is 25C with standard deviation of 5C. When the sensed temperature exceeds 28 c, the message that contains this data is considered a high priority message. Common nodes and cluster head nodes aggregates produced data and received messages, respectively, and disseminate the aggregated message periodically.

In order to simulate communication problems, at each 20 seconds, common nodes have their sensing and disseminating interval decreased from 0.01 and 1 seconds to 0.001 and 0.01 seconds, promoting an increase in network data production. As a consequence to that, messages are lost because of space loss at the packet queue, that admit 10 messages in common nodes and 100 messages in cluster head nodes.

The network nodes try to detect incidents at every 1 second interval. In addition to that, common nodes send messages to update the cluster head knowledge base at each 1 second interval.

6.1 Results

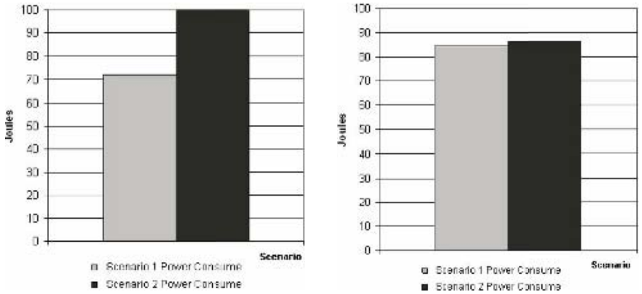
This section presents the simulation results and their evaluations.

Power Consumption

Energy is the resource responsible by the functioning of all sensor nodes modules. In case of nodes failure due to energy exhaustion, the network production is reduced in an irreversible manner.

Sensor nodes from scenario 2 get out of service after 75 seconds of simulation, supporting the communication problem event for a 25 seconds period. On the other hand, the common nodes of scenario 1, which implements the self-management system described in section 5 survive during the entire simulation, enduring the communication problem for a 60 seconds period, in which nodes detect incidents and adapt themselves in order to keep the service available.

The graphic of Figure 4(a) presents the power consumption in the common nodes for both simulation scenarios.



(a) Common nodes power consumption.

(b) Cluster head nodes power consumption.

Fig. 4. Network Power Consumption

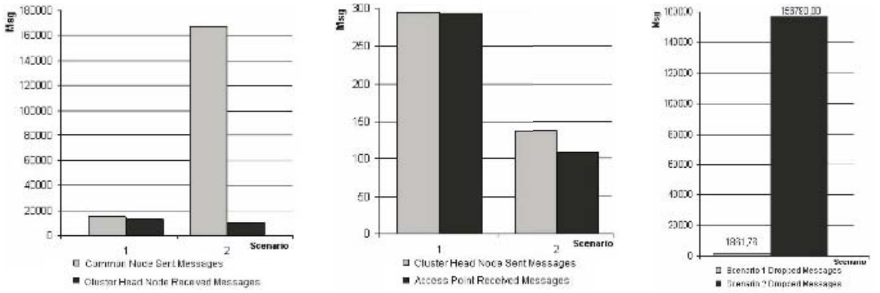
The power consumption of scenario 1 (see Figure 4(a)) is smaller than the power consumption in scenario 2, even though the network keeps functioning for an 80 seconds period larger than the scenario 2. This happened because nodes detect that messages are being lost and diminish their production. Besides, the fact that common nodes disseminate only priority information, in case of low energy level, promotes power saving.

The cluster head nodes from both scenarios presented a similar power consumption (see Figure 4(b)). Although the cluster head nodes from scenario 1 delivered more information, the size of the messages was smaller, making the power consumption comparable to the ones of scenario 2.

Sensed Data Flow

Figures5(a) and 5(b) presents the sensed data flow in the network. The amount of data sent by the common nodes of scenario 1 is 10,80 times greater than the one of the scenario 2 (see Figure 5(a)). However, the amount of data received by the cluster head nodes from scenario 1 is 1,3 times greater than the one of scenario 2. It means that the scenario 1 saves energy, since it delivers and drops less data because it tries to deliver messages when the network is not congested, and, even though, the number of messages delivered to the leader node is bigger if compared to the one of scenario 2. This demonstrates that the number of dropped messages in the network was diminished.

The amount of data sent by the cluster head nodes to the access point in scenario 1 is 2,16 times greater than the one in scenario 2 (see Figure 5(b)). This happens because the leader nodes decide to decrease their dissemination intervals, being able to deliver the received messages by the common nodes of the cluster. Even though, the scenario 1 presented only 0,40% of message loss, while the scenario 2 presented 21,07% of message loss.



(a) Common nodes sent messages and cluster head nodes received messages.

(b) Cluster head nodes sent messages and access point received messages.

(c) Network dropped Messages.

Fig. 5. Network sensed data flow and message drop

The amount of dropped messages is presented in Figure 5(c). The scenario 2 drops number is 84,22 times greater the scenario 1, because it is not implemented any way of detecting communication problems nor either recovering from these problems.

Analyzing all the obtained results, the proposed solution has proved to be efficient in correcting communication problems and prolonging the network lifetime.

7 Related Work

It is notable the WSN area progress. However, in the scientific view, these networks present a great variety of new problems not studied yet or still incipient.

This is the case when it is considered the WSNs as information technology systems, or even when it is intended to define these networks as autonomic systems. For this reason, this work aims to be a contribution to the area when it deals with the challenge of applying the practice of ITIL processes and autonomic computing to WSNs. To the best of our knowledge, there is no directly related work to the use of ITIL in WSN, so this section presents some existing works on autonomic systems and autonomic WSNs.

An autonomic architecture to WSNs was proposed in [10]. The Autonomic-Oriented Architecture (AoA) was defined to support WSN self-organization. This architecture is allows a dynamic and intelligent control of networks built through intelligent software agents that offer reliability, quality of service, security, and mobility management.

The utilization of autonomic computing to perform symptom analysis is described in [8]. The main contribution of this work is the definition of a symptom model and its components (symptom artifacts and relationships). In [9] the author presents some IT scenarios that benefit from a symptoms-based autonomic computing architecture such as security, service support, service availability and service continuity.

The work [13] considers self-healing aspects in autonomic networks, in particular, techniques for events correlation and fault identification. The authors proposed an environment that performs problem determination and rule discovery for fault identification in telecommunication systems using alarm events.

Some autonomic functionalities were implemented for a WSN in [12]. This work defines policies to the accomplishment of the self-organization, self-awareness, self-knowledge services and service negotiation. The results showed that implementing autonomic functionalities and service negotiation benefits the energy saving and the quality of the information extracted from the network.

8 Conclusion

This work considers WSNs as IT tools or systems, since they produce, process, store and deliver data. As an IT system, the WSNs are based on the following components: sensor nodes hardware and software, network elements communication, and network produced information management. This work also proposes that WSNs should be designed as autonomic systems that implement different management services, such as self-organization, self-configuration, self-diagnosis and self-healing.

To the best of our knowledge, this is the first work in literature that considers the use of ITIL best practices to autonomic WSNs. The results from the scenarios, which implement the service management system for self-healing WSNs, show that this can be a great solution to manage failures, considering that these are not exceptions in this kind of network.

The self-healing system designed as study case deals with the specification of the monitor, analyze, plan and execute autonomic manager functions, which were modeled according to some of the ITIL practices.

The results show that the detection of improper operations and components failure in WSNs and the automatic recovering of problems promote a greater availability of the service and the network longevity.

The autonomic WSNs promote the independence of human intervention in tasks of maintenance and management, reducing the communication costs, the response time to events and increasing significantly the network availability. The design and implementation of autonomic WSN represent a new research opportunity and, specially, when these networks are under the IT paradigm.

References

1. Hochstein, A., Zarnekow, R., and Brenner, W. (2005). Itil as common practice reference model for it service management: Formal assessment and implications for practice. *IEEE International Conference on e-Technology, e-Commerce and e-Service.*
2. IBM (2005). Autonomic computing white paper - An architectural blueprint for autonomic computing.
3. Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *IEEE Computer*, 36(1):41–50.
4. MICA (2003). *MICA Wireless Measurement System.* Crossbow Technology Inc. Available at http://www.xbow.com/
5. Office of Government Commerce (2000). *ITIL Service Support.* The Stationery Office.
6. Office of Government Commerce (2001). *ITIL Service Delivery.* The Stationery Office.
7. Office of Government Commerce (2002). *Planning to Implement Service Management Manual.* The Stationery Office.
8. Perazolo, M. (2005a). Symptoms deep dive, part 1: Know thy symptoms, heal thyself. IBM Autonomic Computing DeveloperWorks.
9. Perazolo, M. (2005b). Symptoms deep dive, part 2: Cool things you can do with symptom. IBM Autonomic Computing DeveloperWorks.
10. Pujolle, G. and Chaouchi, H. (2005). An autonomic oriented architecture for wireless sensor networks. *Annals of Telecommunications - Sensor Networks*, 60(6/7):819–830.
11. Ruiz, L. B. (2003). *MANNA: A Management Architecture for Wireless Sensor Networks.* PhD thesis, Computer Science Department of the Federal University of Minas Gerais, Belo Horizonte, MG, Brazil.
12. Ruiz, L. B., Braga, T. R. M., Silva, F., Assunção, H. P., Nogueira, J. M. S., and Loureiro, A. A. F. (2005). On the design of a self-managed wireless sensor network. *IEEE Communications Magazine*, 43(8):95–102.
13. Sterritt, R. (2004). Autonomic Networks: engineering the self-healing property. In *Journal of Advanced Engineering Informatics, Engineering Applications of Artificial Intelligence*, volume 17, pages 727–739 187. Elsevier Publishers.
14. WINS (2002). Wireless Integrated Network Sensors (WINS). Available at http://www.janet.ucla.edu/WINS/.
15. Lopes, C. E. R., Melo, J. C., Assunção, H. P., Braga, T. R. M., Silva, F. A., Ruiz, L. B., Loureiro, A. A. F. and Nogueira, J. M. S, (2006). MannaSim: Simulando Redes de Sensores Sem Fio. *24th Brazilian Symposium on Computer Networks (SBRC'06).*

Integration of Mobile IPv6 into Mobile Ad-Hoc Network Systems

Monden Kazuya[1], Satoh Hiroki[1] Yamamoto Junji[2], Shomura Yusuke[2], Shimizu Atsushi[1], Hayashi Masato[1], Matsui Susumu[1], and Yoshizawa Satoshi[2]

[1] Hitachi, Ltd, Systems Development Laboratory
1099, Aso-ku Ozenji, Kawasaki, 215-0013, Japan
{monden, h-satoh, shimizu, m-hayash, matsui}@sdl.hitachi.co.jp
[2] Hitachi, Ltd, Central Research Laboratory
1-280, Higashi-Koigakubo, Kokubunji-shi, Tokyo 185-8601, Japan
{junji-y, yusuke-s, yoshi}@crl.hitachi.co.jp

Abstract. Mobile ad-hoc network (MANET) technology is a wireless network system that may make ubiquitous computing possible. This technology is especially promising for vehicle-to-vehicle and road-to-vehicle communication. However, there are some problems with IP mobility support in MANETs. One technology that provides IP mobility to mobile nodes is Mobile IP. To solve IP mobility problems in MANETs, we examined how Mobile IP technology can be integrated into MANET.

1 Introduction

Mobile ad-hoc network (MANET) technology, which constructs wireless multihop network autonomously without any infrastructure, is a wireless network system that may make ubiquitous computing possible. In intelligent transport system (ITS) services, a MANET enables vehicle-to-vehicle (V2V) and road-to-vehicle (R2V) communication. In V2V communication, vehicles communicate directly with each other without any infrastructure such as base stations. In R2V communication, vehicles communicate with road nodes via gateways on the road.

One application of V2V communication is a vehicle control system that prevents rear-end accidents, based on urgent information, such as a sudden braking, sent from a vehicle in front using V2V communication. One example of R2V communication is vehicle diagnosis using vehicle relationship management [1] technology.

Using V2V communication, if a MANET is composed of vehicles, the route between them is constructed dynamically and a node can communicate with another node outside of its wireless range using multiple hops, without an infrastructure. MANETs are better at dynamic routing than other systems and do not require infrastructures. Using R2V communication, a server node in a network with an infrastructure, like the Internet, can connect with a mobile node via the network. For example, if a MANET is composed of vehicles and a gateway to the network, vehicles can communicate with the server via the

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 229–242, 2006.

gateway. When a mobile node moves and connects to a new network via a gateway, it has to acquire a new IP address belonging to the network. However, the node cannot keep communicating with correspondent nodes using the previous IP address. This means that you cannot keep talking with your friends using IP telephony via the Internet while driving. For ubiquitous computing, the talk should continue even if the node that you are using connects to a new network. To accomplish this, Mobile IP [2] [3] has been proposed. Mobile IP is designed to provide IP mobility to mobile nodes over the Internet. Mobile IPv6 [3] is especially promising because it can handle the number of nodes needed in the ITS services that we investigate. Mobile IPv6 has several benefits: First, a mobile node can keep communicating with correspondent nodes using its own unique IP address. Second, the network that the mobile node is connected to can be changed automatically. Therefore, integrating Mobile IPv6 into a MANET is effective in V2V and R2V communication. However, combining these two technologies presents some problems. One problem is that new IP addresses cannot be acquired using multiple hops. Therefore, we integrated Mobile IPv6 into a MANET to achieve seamless roaming between networks. We will propose our method for combining Mobile Ipv6 and MANETs in Section 4.

2 MANET and Mobile IPv6

In this section, we introduce two important technologies: MANET and Mobile IPv6.

2.1 MANET

A MANET is a collection of mobile computing devices (network nodes) connected by wireless communication technology. The network has three main features: it is established by instant autonomous networking, it is multi-hopping, and its topology is dynamic. First, nodes discover their neighbors by periodical beaconing, which allows each node to exchange information (node attributes like IP addresses, IDs, information on its neighbors, etc.) with its neighbors to establish connections automatically. Second, if a destination node moves outside radio range of a source node, intermediate nodes help relay data to the destination. Third, when nodes move during communication, each node automatically searches for new neighbors and calculates a new route between a source and its destination, which is called dynamic routing.

Routing protocols fit into two major categories: proactive and reactive. Optimized link state routing (OLSR) protocol [4] is a proactive protocol that each network node uses to periodically exchange topology information with other nodes so that routing tables are ready before nodes send packets. Ad hoc on-demand distance vector (AODV) [5] is a reactive protocol that a source node uses to broadcast route query packets based on requests to send information, and when intermediate nodes receive these packets, they create reverse route entries to the source node. When the destination node receives the query packet, it sends a route reply packet to the source node. This packet gives the order of intermediate nodes. Therefore, when a node receives a route reply packet, it makes a routing table entry for the destination node and sends the requested information. OLSR has a shorter delay in sending data than an AODV, but it constantly consuming wireless bandwidth. An AODV has lower routing overhead than OLSR, but taking the time needed to make routes delays transmission. An IETF [1] working group known as MANET WG [2] has endeavored to standardize these two protocols.

We use OLSR as our MANET protocol because, in view of time-sensitive applications such as vehicle control systems, a proactive routing protocol that has less delay is better.

2.2 Mobile IPv6

Mobile IPv6 is designed to allow nodes to be reached and maintain ongoing connections while changing their locations. Mobile IPv6 provides a function similar to post office forwarding to provide network connectivity to mobile nodes. In Mobile IPv6, each mobile node has a unique IP address, called a home address (HoA). A HoA is permanently assigned to each mobile node within its home subnet prefix on its home link. Each mobile node has a home agent on its home link with which it registers its temporary care-of address (CoA) on the foreign link it is attached to. When a mobile node moves to a foreign network, it receives a router advertisement (RA) packet from a router in the foreign link and therefore, knows it has moved. Its CoA is generated by replacing the node's subnet prefix with the foreign subnet prefix in the RA. The mobile node registers its CoA with its home agent (HA) by sending a binding update message to the HA. The HA replies to the mobile node with a binding acknowledgement message. After finishing this procedure, the HA intercepts packets on the home link destined for the mobile node's home address, encapsulates them, and transmits them to the registered CoA.

3 Problems in Integration

This section describes problems in integrating Mobile IPv6 into a MANET.

3.1 Acquiring Care-of Addresses

Using Mobile IPv6, a mobile node acquires a foreign subnet prefix by receiving an RA from a router in the network to which it has moved, and then a CoA is generated. Although the RA from the router is transmitted to the all-node multicast address (ff02::1) [6], a multicast packet cannot be forwarded by intermediate nodes because the standardized MANET routing protocols are unicast routing protocols. Therefore, the RA does not reach mobile nodes connected to

[1] Internet Engineering Task Force http://www.ietf.org

[2] Mobile Ad-Hoc Networks(manet) http://www.ietf.org/html.charters/manet-charter.html

the router via more than one hop, so that the mobile nodes cannot detect their movements to the foreign link and acquire CoAs. So that they can acquire CoAs, we must develop a method of distributing network prefixes to mobile nodes that are connected to the router with multi-hop wireless links.

3.2 Routes to CoA in MANETs

Next, we describe the problem of making routes to CoAs. Using Mobile IPv6, a mobile node that acquires a CoA sends a binding update message to its HA. The HA returns a binding acknowledgement message to the mobile node. To accomplish this, interactive communication must be possible between the mobile node and its HA. The binding update packet must reach the mobile node's HA and the binding acknowledgement packet must reach the mobile node's CoA. This means that a route must be constructed between the mobile node's CoA and HA in the MANET. In OLSR, the route can be constructed by a host and network association message, which is an OLSR control message. The message is used to provide connectivity between the MANET and external networks. However, no standardized way has been developed to make CoAs into routable addresses on MANETs.

We now propose a way to construct routes to mobile nodes' CoAs in a MANET.

3.3 Redundant Routes to HoAs

Nodes communicate using their HoAs. Packets are transmitted to HoAs when mobile nodes are attached to links away from their home links, and the packets are encapsulated in packets destined for the nodes' CoAs by the HAs and delivered to the nodes. Even if two nodes in a MANET can communicate with each other without messages passing through their HAs, packets are still transmitted via the HAs. These routes are redundant. Therefore, the delay and the waste of network resources, such as wireless bandwidth, are higher than a system in which communication is done only within the MANET. For example, in a vehicle network, two vehicles connected by multiple hops cannot communicate with each other directly in a MANET without messages passing through the Internet. This means that communication via HAs is not suitable for the time-sensitive applications. Therefore, a communication route that does not pass through the Internet must be constructed.

3.4 Multiple Wireless Media

The combination of MANETs and Mobile IPv6 allows mobile nodes to expand the area where the Internet can be accessed via gateways. However, if there is no intermediate node between a mobile node and a gateway, the mobile node cannot communicate via the Internet. This means that we have not yet achieved ubiquitous computing. If a mobile node has multiple wireless interfaces such as a wireless LAN (WLAN) device and a cellular phone and uses them depending on which is available, it will communicate using the WLAN where it can communicate with a destination node via a MANET and communicate using the cellular phone where it cannot communicate with a destination node via a MANET. In this case, we assume that MANETs can only be accessed using WLAN interfaces because cell phones cannot directly connect mobile nodes without base stations. Therefore, the problem is how to change between the cellular networks and MANETs.

4 Solutions to the Problems

In this section, we propose solutions to the problems we clarified in the previous section. We focus on IPv6-based networks. Therefore, we have to extend OLSR to support IPv6. In IPv4, OLSR control messages are broadcast, but IPv6 has no broadcast address. Control messages are transmitted to the all-node multicast address (ff02::1), in the same way that an RA is advertised. Moreover, we added the following extensions for the integration.

4.1 Method of Distributing Network Prefixes

An RA multicast from a router does not reach mobile nodes connected to the router with multiple hops. However, OLSR control packets are flooded to all nodes even if they are connected with multiple hops. This flooding is multipoint relay (MPR) flooding, which is more efficient than traditional flooding. A comparison of traditional flooding and MPR flooding is shown in Figure 1. MPRs are special nodes that transmit control messages and are selected by source nodes to reduce retransmission of the messages. Clausen and Jacquet (2003) explain a selection scheme for MPRs [4]. We focus on this efficient flooding scheme. To distribute the foreign subnet prefix to all nodes that can be connected with the router, the packet that stores information on the prefix is transmitted as an OLSR control packet. We call this control packet an RA/OLSR. A mobile node that receives an RA/OLSR can acquire its subnet prefix and generate a CoA.

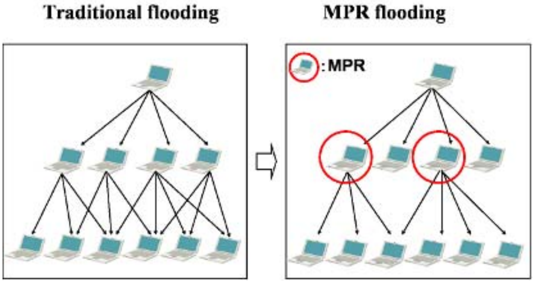


Fig. 1. Comparison of flooding schemes

The gateway between a MANET and a traditional IP network acts as a router and sends an RA/OLSR. First, a MANET must be established to deliver an RA/OLSR including the foreign subnet prefix. For that, we assign a MANET address, which is used to establish the MANET among mobile nodes.

Now, mobile nodes have three IP addresses.

1. Home address: the permanent address of a mobile node. This address is within the home network of the node. Applications communicate with each other using their home addresses.
2. Care-of address: generated by an RA/OLSR. It is used to communicate with the HA.
3. MANET address: Address used to establish a MANET.

4.2 Construction of Routes to CoAs

When a mobile node gets a CoA, it performs a binding update message and receives a binding update acknowledgement. A bidirectional route to the mobile node's HA is necessary to accomplish this procedure. OLSR can provide a route to the mobile node's home link, but a return route must be made to the CoA from HA. The route from the home link to a gateway is constructed in traditional IP networks, so we do not need to construct it. However, a route from the gateway to the mobile node is not provided by MANET routing protocols. In the following, we focus on an OLSR function. OLSR allow mobile nodes to have multiple wireless interfaces that are assigned routable addresses in MANET. This function allows a mobile node to have multiple routable addresses. We have designed a method of assigning MANET addresses and CoAs to the multiple routable addresses. When a mobile node acquires a CoA, we assume that the mobile node got a new wireless interface that was assigned the CoA, or an old interface was reassigned the CoA. Then, the mobile node begins to advertise the CoA as its multiple routable address. Therefore, the CoA becomes a routable address, and OLSR constructs a route to the CoA in the MANET. This method provides a bidirectional route between a mobile node and its home link.

4.3 Optimized Route

In Mobile IPv6, packets sent from a mobile node to another mobile node are transmitted first to the HA and then are sent to the destination node after the HA changes the destination of the packets to the CoA from the HoA (Figure 2, route (A)). Even if the destination node is a neighbor, this mechanism carries packets to the destination node through a redundant route. Mobile IPv6 has a function that optimizes routes: the source node sends packets directly to the destination node's CoA (Figure 2, route (B)) However, the source node must send a packet to the destination node's HA at least once to know the destination node's CoA. Moreover, the mobile nodes (MNs) are assumed to have two wireless interfaces such as one MANET interface (a WLAN) and one non-MANET interface (a cellular phone), the MNs are assumed to connect with the gateway (GW) using the cellular phones, and the two MNs are assumed to connect using the WLAN. The shortest path between the MNs based on the scheme is not route (B) but MN1 → GW → MN2 because, in this case, MN's CoA is assigned from the gateway to a cellular network, so it is not a MANET routable address. This optimization scheme is not suitable for time-sensitive communication. Therefore, we propose a new route optimization scheme to integrate Mobile IPv6 into MANET. We focus on a mobile nodes' routing tables. If there is a host route to MN2 in MN1's routing table, MN1 sends packets to MN2 directly because host routes are prioritized over network routes. In reality, the MANET address and CoA of every MN are registered in each MN's routing table as host routes. If the HoA of an MN can be registered to a routing table as a host route, the route can be optimized. The route can be optimized using a method similar to the construction of the route to the CoA described in the previous paragraph. A HoA becomes a routable address if a mobile node has a virtual interface that is assigned the HoA. After that, OLSR makes a route to the HoA in the MANET so that packets to the HoA in same MANET are transmitted to the corresponding node directly (Figure 2, route (B)).

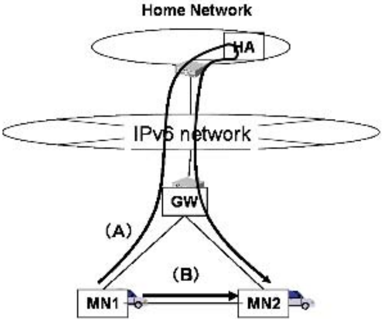


Fig. 2. Redundant route and optimized route

4.4 Selection Method of Wireless Media

We focus on mobile nodes that have two wireless devices; one device is a WLAN and the other is a cellular phone. A MANET is established using the WLANs; the cellular network is not a MANET because a cellular phone cannot communicate directly without a base station. We think that a cellular network can complement a MANET. If the mobile node can communicate with its destination node using either interface, it should use the WLAN because of its higher bandwidth and lower communication costs. Available networks, including MANET and cellular networks, can be detected by receiving RAs and RA/OLSRs. Therefore, the interface used to communicate is switched based on the reception or time-out of RA/OLSRs.

5 Experimentation

We implemented the solutions developed in Section 4 to construct the system shown in Figure 3.

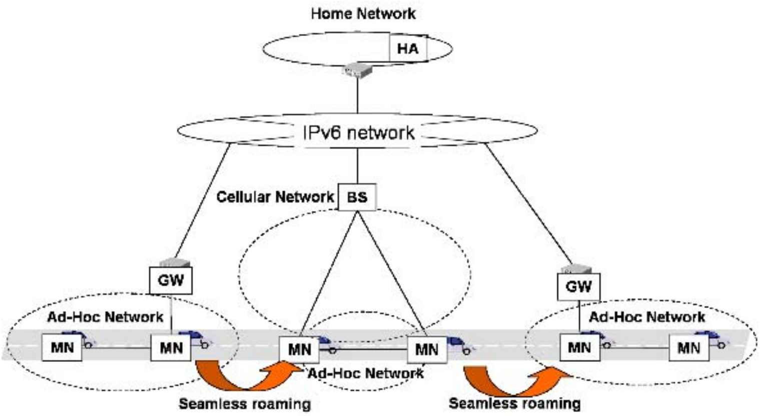


Fig. 3. Structure of network

5.1 Structure of Experimental System

Each mobile node has two communication interfaces: a wireless LAN interface and a cellular phone. Extended OLSR is installed on each MN. MANETs use only wireless LAN. Gateways connect the MANETs and an IPv6 wired network. The gateways have two interfaces: wireless LAN and wired LAN. The gateways periodically send RA/OLSRs as routers. The MNs seamlessly communicate with destination nodes using their wireless LANs and cellular phones. A HA is in each MN's home network. A MN's home network, gateways, and base stations (BSs) of cellular networks are routable with each other. The MN communicates seamlessly by switching between MANETs and cellular networks.

5.2 Applications

We used video streaming between the mobile nodes as an example of a service. One application of the streaming is the Video Conferencing Tool (VIC)[3]. We developed software to monitor nodes' available routes and the positions of the nodes based on global positioning system (GPS) information.

5.3 Experimental Conditions

We prepared two laptops as MNs and put them in vehicles (Figure 4). Each MN's wireless LAN device was IEEE802.11b. We attached external antennas to wireless LAN devices and GPS device on the roof, as shown in Figure 5. MN1 had a camera for video streaming. The vehicles went around a circuit of about

[3] http://www-nrg.ee.lbl.gov/vic/

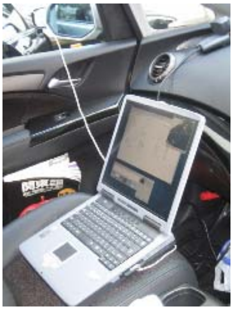


Fig. 4. Mobile node in a vehicle

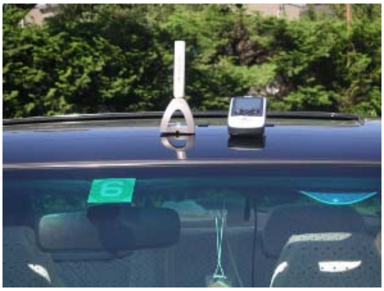


Fig. 5. Left:external antenna Right: GPS device

2 km (Figure 6) at 40 km/h. Two gateways (GW1 and GW2 in Figure 6) were located near the circuit. The gateways were connected to the IPv6 Internet using asymmetric digital subscriber lines (ADSL). The routes available to each node were checked on MN2 using the monitoring software we developed. The quality of the communication was tested by examining the quality of a video stream of VIC transmitted from MN1. The feasibility of the communication was done by available routes and GPS information transmitted from other nodes including GWs. The vehicles went around the track twice. On first lap, the two vehicles drove together. On second lap, MN2 stopped near GW1, and MN1 went around alone. We tested the following cases.

Case 1. A MN connected to a gateway between a MANET and the Internet by multiple-hops was sent an RA/OLSR. We checked that the MN received it.
Case 2. A MN generated a CoA and sent a binding update. We checked to see if a binding acknowledgement was received.
Case 3. The path between MN1 and MN2 was tested to see if it was the shortest.
Case 4. We checked that MNs switched communication interfaces to the Internet as available and continued communicating with destination nodes.

6 Results

The following figures are screen shots captured on MN2. First, we will explain the screen elements. In Figure 7, the upper left window is the software that monitors the node's available routes and its position. Each node is a box, and MNs are cars. The node labeled "Relay" was a fixed relay node, placed where it is because GW1 was not close enough to the circuit. The lines show available routes. Lines between GWs and the Internet show that two gateways are connected via the Internet using ADSL. Lines between cars and the Internet show a direct connection to the Internet using a cellular network or a connection between MNs in a MANET. Every node sent information, such available routes and its routable address, to MN2's HoA. The upper right window shows video streaming from MN1. Under the left window is a message application, similar to a chat,

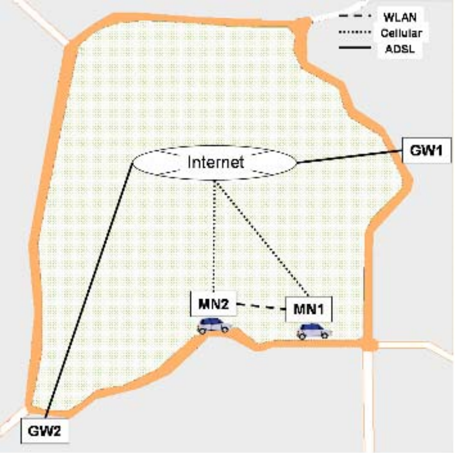


Fig. 6. Experimental circuit

and it shows urgent messages from MN1. Under the right window is a control terminal. Blue and red balls show routes between MN1 and MN2; blue balls show routes from MN2 to MN1, and red balls show return routes from MN1.

6.1 Case 1 and Case 2

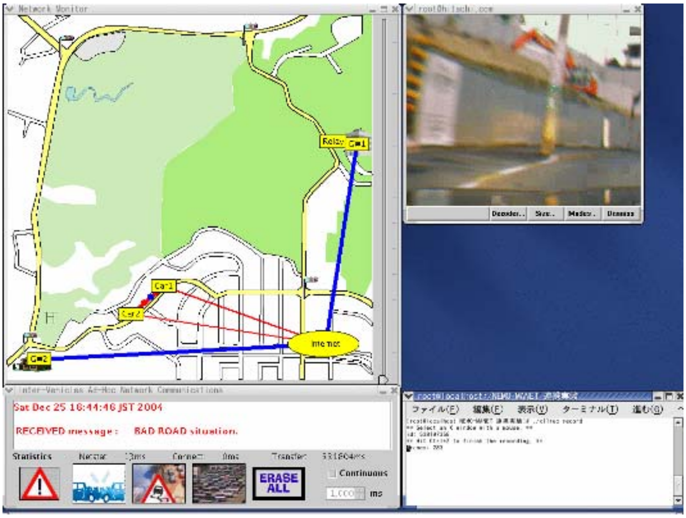
MN2 was connected to the Internet by multiple hops, generated its CoA, and constructed a route to its CoA. We know this because the receipt of GW1's and Relay's routing information packets by MN2 was displayed in the monitor window (Figure 8). An RA/OLSR was relayed by MN1. MN2 generated its CoA, sent a binding update, and received a binding acknowledgement. This means that the tests of cases 1 and 2 were successful.

6.2 Case 3

MNs chose optimized routes even if they could choose other routes via GW1 and cellular networks, as shown in Figures 7 and 9. Therefore, our optimizing method works well.

6.3 Case 4

The MNs connected to the Internet via a MANET, and the route between them was in the MANET, as shown in Figure 9. Figure 10 shows a later time than Figure 9. MN2 stopped near GW2. MN1 kept driving and changed the interface it used to access the Internet from its wireless LAN to its cellular phone. The media was changed because MN1 didn't receive an RA/OLSR from GW1. After the moment shown in Figure 10, MN1 connected to GW2, as shown in Figure 11. This figure shows that MN1 changed the interface to its wireless LAN after receiving an RA/OLSR, so MN1 and MN2 continued communicating. These

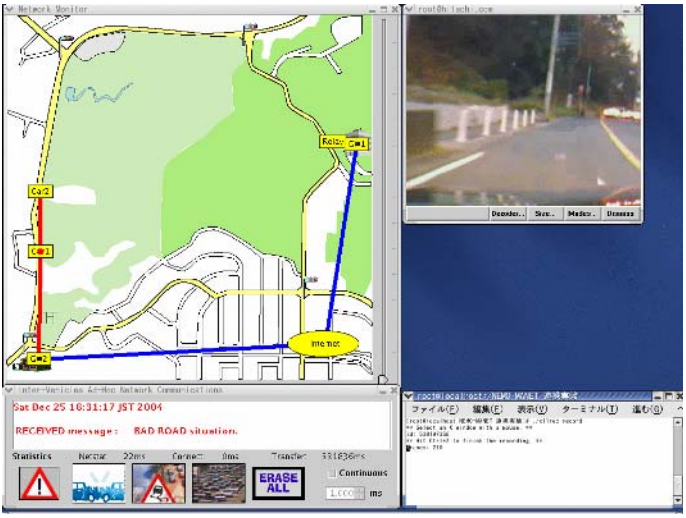


Sat Dec 25 16:44:46 JST 2004

RECEIVED message : BAD ROAD situation.

Statistics Nectar 13ms Connect 0ms Transfer 331804ms

Fig. 7. MN1 →MANET →MN2



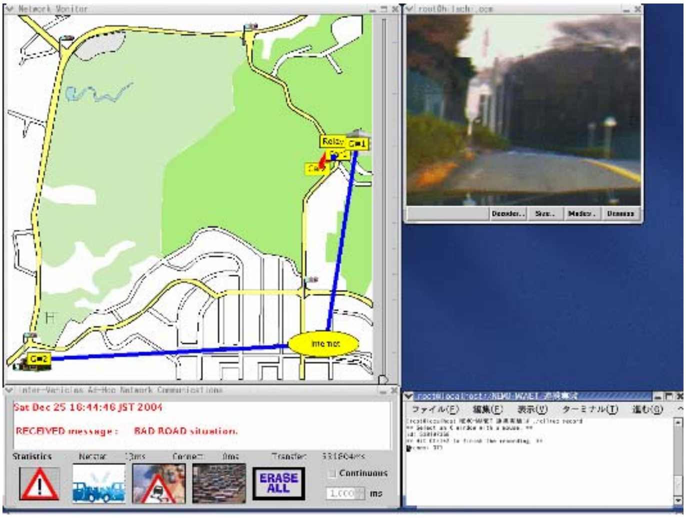
Sat Dec 25 16:31:17 JST 2004

RECEIVED message : BAD ROAD situation.

Statistics Nectar 22ms Connect 0ms Transfer 331836ms

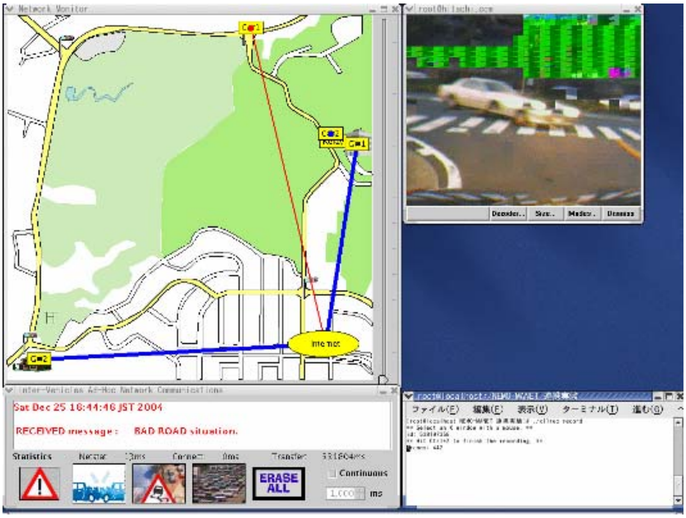
Fig. 8. Multi-hop route to Internet via gateway

three figures (9, 10, and 11) show that MN1 changed the interface while driving, but the communication continued. This means that our media selection method allow MN1 to seamlessly change its interface. However, at the time that the network was switched, the communication was cut off. We believe the reason is that the available networks had already changed to only the cellular network by the time the RA/OLSR timed out.


Sat Dec 25 16:44:46 JST 2004

RECEIVED message : BAD ROAD situation.

Fig. 9. MN1 →MANET →MN2

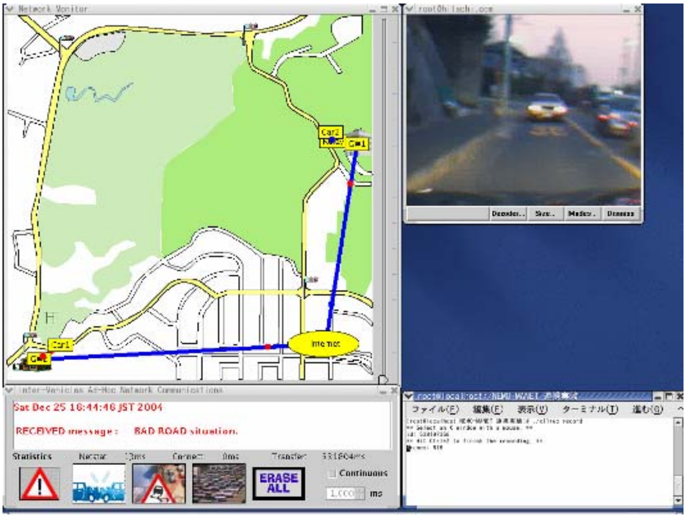

Sat Dec 25 16:44:46 JST 2004

RECEIVED message : BAD ROAD situation.

Fig. 10. MN1 →Cellular → Internet → MANET →MN2

6.4 Communication Quality

The picture from MN1 in Figure 10 has block noise, but the other screen captures have no block noise. Previous routes from MN1 to MN2 passed through a cellular network. Therefore, one reason for the difference of video quality is that the cellular network had a narrower bandwidth than the wireless LAN. MANETs


Network Monitor
Car2
Car1
Internet
Decoder.. Size.. Modes.. Dismiss
Inter-Vehicles Ad-Hoc Network Communications
Statistics
ERASE ALL
Continuous
1,000 ms
ファイル(F) 編集(E) 表示(V) ターミナル(T) 進む(G)

Sat Dec 25 18:44:46 JST 2004

RECEIVED message : BAD ROAD situation.

Fig. 11. MN1 →MANET →Internet → MANET→MN2

enabled the broadband area to be extended, Mobile IPv6 provided IP mobility to MNs, and the cellular network complemented the MANETs.

7 Conclusion

We studied the integration of Mobile IPv6 into a mobile ad-hoc network system and came to the following conclusions.

1. Mobile nodes in a MANET can be connected to the Internet with foreign subnet prefixes that a gateway between the Internet and the MANET sends, included in an RA/OLSR.
2. Advertising a CoA as a multiple routable address allows mobile nodes to create bidirectional routes to their home links.
3. Routes can be optimized by constructing host routes to destination nodes' HoAs in a MANET.
4. Detection of MANET availability based on receiving and time-out RA/OLSRs allowed MN1 to change its interface.
5. Integration of Mobile IPv6 into a MANET can expand the range of wireless LAN and enables mobile nodes to communicate seamlessly when they move to new networks.

This integration system can optimize routes, reduce delay, and lower communication costs according to the destination of a packet. This system is a necessary network for ubiquitous computing. The system can be applied not only to ITS systems, but also to other wireless communication systems. Next, we will study the possibility of reducing the period in which communication is impossible by changing the network and extending Mobile IPv6 to network mobility [7].

References

1. Aizono, T., Endo, Y., Otsuji, S., Yamane, K., Shimura, A.: Hitachi groups initiatives regarding trends in vehicle information systems. HITACHI REVIEW **53** (2004) 229–235
2. Perkins, C.: RFC 3220: IP Mobility Support for IPv4 (2002) http://www.ietf.org/rfc/rfc3344.txt.
3. Johnson, D., Perkins, C., Arkko, J.: RFC 3775: Mobility Support in IPv6 (2004) http://www.ietf.org/rfc/rfc3775.txt.
4. Clausen, T., Jacquet, P.: RFC 3626: Optimized Link State Routing Protocol (OLSR) (2003) http://www.ietf.org/rfc/rfc3626.txt.
5. Perkins, C., Belding-Royer, E., Das, S.: RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing (2003) http://www.ietf.org/rfc/rfc3561.txt.
6. Narten, T., Nordmark, E., Simpson, W.: RFC 2461: Neighbor Discovery for IP Version 6 (IPv6) (1998) http://www.ietf.org/rfc/rfc2461.txt.
7. Devarapalli, V., Wakikawa, R., Petrescu, A., Thubert, P.: Internet-draft: Network Mobility (NEMO) Basic Support Protocol (2004) http://www.ietf.org/internet-drafts/draft-ietf-nemo-basic-support-03.txt.

AToM: Atomic Topology Management of Wireless Sensor Networks

S. Shen[1], G.M.P. O'Hare[1], D. Marsh[1], D. Diamond[2], and D. O'Kane[1]

[1] Adaptive Information Cluster
School of Computer Science and Informatics
University College Dublin, Belfield, Dublin 4, Ireland
{song.shen, gregory.ohare, david.marsh, donal.okane}@ucd.ie
[2] School of Chemical Sciences
Dublin City University, Dublin 9, Ireland

Abstract. This paper explores the structural and behavioural similarities that exist between the chemical constitution of natural substances and WSN (Wireless Sensor Network) topology. It introduces Atomic Topology Management (AToM), which uses concepts like WSN electron, WSN nucleus, WSN photon, and WSN atom to model sensor node, base station, message and basic WSN subset consisting of one base station and several sensor nodes. We subsequently extend the modeling to explain the basic behaviours of WSNs. The paper describes naming rules for topology management and based upon this, we devise and test some basic algorithms for energy and load balancing.

Keywords: Wireless Sensor Network, Topology Management, Autonomic Networking.

1 Introduction

Recent technology improvements in wireless communications and electronics have enabled the development of low-cost, low-power, multi-functional sensor nodes which possess the potential for self-organization and which can opportunistically collaborate in order that they may more effectively achieve, and continue to achieve, their overall goal. Large-scale sensor applications can be realized with such densely distributed micro nodes [1] [2].

Micro sensors of this nature are inherently resource-bounded, specifically with regard to energy (usually supplied by AA batteries). Such sensor nodes can typically merely communicate directly with those neighbours within a limited distance, rather than communicating with a Base Station (BS). A typical Wireless Sensor Network (WSN) consists of one or several BSs together with a large quantity of sensor nodes which sense the environment and work collaboratively, processing and routing the sensor data.

Little consideration has, as yet, been given to the issues of proper naming and topology management [13] [14] without which, management and analysis of the basic behaviors of a WSN are difficult. Several basic issues, such as the management of

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 243 – 254, 2006.

sensor network, coverage of sensor nodes [11], load balancing of neighbour subsets of WSN [3] [4] and routing for event data [8] [9] [10] [12], still remain open issues. These issues are closely related to self-configuring (or more generally topology management) of WSNs, since wireless communication is vulnerable to environmental changes. As sensor networks scale up in size, effectively identifying and managing the distribution of the network load will become of paramount importance. Creating a naming rule and building up a management model could well address and simplify research problems on architectural management and routing. However, modelling topology and the basic behaviours of sensor networks is inherently difficult because unlike other computer networks, the basic communication behaviours of the wireless sensor nodes can be described by a series of relayed discrete hops, each of which consumes approximately the same amount of energy. Within a WSN subset, a base station often serves as the data aggregation point or the sink of the data in the network, while those sensor nodes scattered around it are considered as resource limited relay points. With power consumption always the top priority, the basic behaviours of WSNs are fundamentally different from other networks.

Within this paper, we investigate possible analogues that may enable the more effective management of topologies of WSNs. We assume that the sensor network in question is not of a fixed nature determined in an a priori manner but rather one that comprises of sensors that are mobile and which potentially can determine when and where to move and/or respond to requests to do so. Therefore the naming and topology management of a WSN will be fundamental to its autonomic behaviours, such as self-configuring, self-healing, self-optimizing, and self-protecting. Fortunately, the world offers many resemblances to WSNs from which we can borrow. In particular, within this work we examine the chemical composition of natural substances. We advocate borrowing from this domain in the management of WSN topologies in this regard this work is pioneering and novel. Furthermore it is our contention that such a model can effectively underpin autonomic decision making and assist in the delivery of self properties. This analogy forms the core component of this paper which seeks to explore the appropriateness of the model in the WSN arena. The analogy helps us primarily in two aspects: one is guiding us to the naming and efficient management of a large topology of sensor networks in a natural way; the other is helping us to find a solution to energy and load balance, which can significantly improve network lifetime and communication ability.

2 A Naming Convention Within Network Architectures

2.1 Definitions

Considering the analogy with chemical compounds we now introduce some WSN definitions which embrace this viewpoint. Within the scope of this paper we consider chemical characteristics are beyond our concern. Therefore, we do not employ concepts such as isotope or ion in our approach.

The core definitions we adopt are now introduced:

Atom: An atom is composed of a nucleus, with electrons orbiting around it. A WSN atom, containing a base station and a subset of sensor nodes, is the smallest functional element within a sensor network just like a natural atom. Without losing generality, we assume that the WSN atom contains one BS.

Photon: A particle of light, associated with a piece of energy needed for an electron to jump to a higher energy level. In WSNs, a photon may be considered to equate to a message flow together with a unit of energy consumption associated with a single hop. The energy for a photon depends on the wavelength of the light; while in the WSN, the energy for a photon is approximately proportional to the length of the message.

Electron: A negatively charged particle orbiting around the nucleus. Within a WSN, an electron may be viewed as analogous to a sensor node distributed around a base station. A WSN electron is also a spot where each hopping message could temporarily reside.

Nucleus: The core of an atom. Within a WSN, the nucleus is viewed as the base station.

Molecule: A molecule, consisting of one or several atoms, is the smallest particle of a compound that has all the chemical properties of that compound. A WSN molecule, consisting of several WSN atoms, is a fundamental application unit of WSNs. A WSN molecule may consist of two or more different types of WSN atoms (Compound), see Fig. 1. The minimum WSN molecule is that of a single WSN atom.

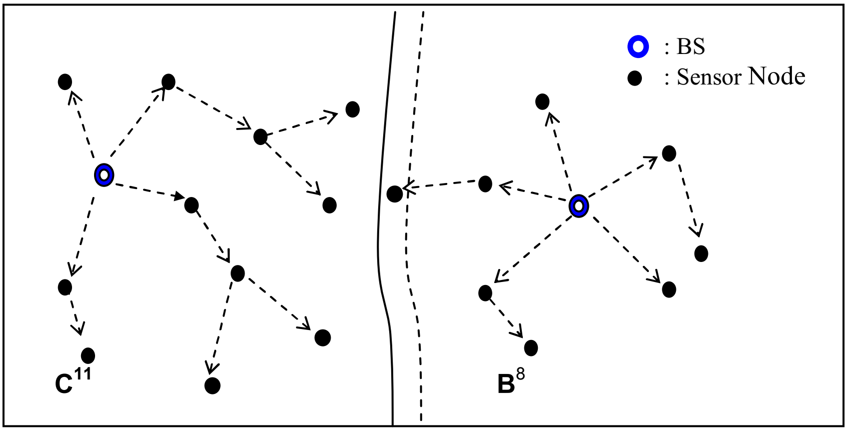


Fig. 1. WSN molecule example

2.2 Similarities and Differences

Based upon these definitions we now explore further the potential benefits of applying this analogy. The atom is an ultimately indivisible particle of an element in nature, while the WSN atom which we use to model a wireless sensor subset, is the smallest existing element of the WSN. On the other hand, a molecule is the smallest functional form of substance, while a WSN molecule, is a fundamental application unit of WSNs. In order to address the process of routing, we calibrate the distance from a given node to the BS into a number of hops, as shown in Fig. 2. Ideally, each hop should be able to move a piece of message onto another node closer to the BS.

Clustering of WSN can be considered as a process of determining each sensor nodes' distance level to BS. The routing of messages toward the BS within a WSN is a process of hopping to a higher level in discrete steps, although the levels may not necessarily be well defined beforehand. Such a process is inherently similar to the jumping of electrons on a higher energy level within atomic structures. The major similarities can be concluded as:

First of all, data from multiple sensor nodes are all required to pass to a particular BS or a few BSs. The BS can be regarded as the centre of a group of nodes, just like the nucleus of an atom. A certain number of sensor nodes together with a BS make a basic unit of WSN subset, which is the smallest usable WSN element, or a WSN atom. The number of sensor nodes within the WSN element together with their energy levels determines characteristics of the WSN element, just as chemical characteristics of an atom is decided by its position on the Periodic Table of Elements. Furthermore, necessities to compose WSN subsets and to balance load on sensor nodes for each WSN subsets or for each level within a subset expedite some kind of distribution rules for sensor nodes. Such composing and balancing behaviours could be modelled by chemical reactions.

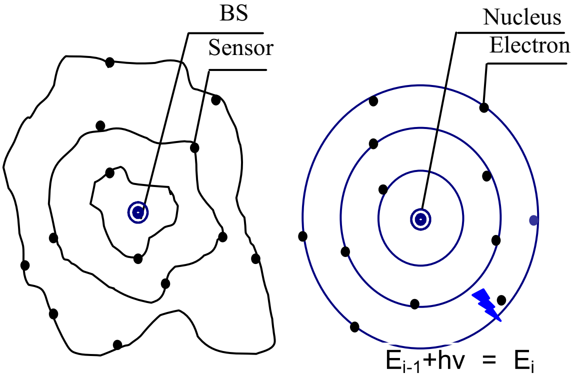


Fig. 2. Modelling of levelled sensor nodes

Second, several different WSN elements can combine into a hybrid structure, like a chemical molecule can be composed of several different chemical elements. The naming convention for such reactions or derived compounds could borrow from that used in chemical reactions.

Third, akin to the jumping of an electron to a higher energy level, each hop of message within a WSN consumes a piece of energy. The energy cost of hopping a message to a higher level is approximately proportional to the length of the message, while energy cost of an electron jump is proportional to the wavelength of the emitted photon. Both are based on the concept of energy level and are independent of the distance of physical movement.

There are also many differences between an atom and a WSN. For example, the constitution of a WSN can have as many levels as necessary and the number of sensor nodes on each level does not have to strictly follow a rule similar to Pauli's Exclusion Principle.

One major difference emerges from the fact that although we simulate sensor nodes as electrons in a WSN subset, they are normally static geographically. What actually hops is the message, rather than sensor nodes themselves. However, our primary concern is clustering related characteristics such as coverage of sensor nodes, the load balancing of neighbouring subsets of WSN, as well as the energy related characteristics such as consumption of message hopping. Therefore, the jumping characteristics of electrons can merely be considered as the flow of a message. In fact, it is not necessary for a WSN to adopt all the characteristics synonymous with the natural structure.

2.3 Naming of WSN Atom, WSN Molecule and WSN Electron

A WSN compound needs to know the structure, constitution and level situation of its WSN atoms for management purposes. Therefore, proper naming of sub items within a WSN is crucial. With the basic terms defined in the preceding section, we can further visually name the structure of WSN atoms and WSN molecules. Such naming could be flexible according to the scale of the WSN and the elaboration necessities. For example, we could use A to denote a WSN atom with only one external level; use B to denote two external levels, C to denote three external levels, and so on. If so, a WSN subset with one BS and 11 sensor nodes distributed in 3 levels could be denoted as C^{11}; a WSN subset with one BS and 8 sensor nodes distributed in 2 levels could be denoted as B^{8}; a WSN molecule composed of WSN atoms B^{8} and C^{11} is named as $C^{11}B^{8}$, while a WSN electron within WSN atom C^{11} can be named as C^{11}, i, j, where i is the level where the electron situates at and j is the sequence number of the electron, see Fig. 1.

Furthermore, if certain sensor placement principles could be formalized for determining the placement of sensor nodes within a given level, it would be possible to define a table of WSN elements, similar to the Periodic Table of Chemical Elements.

3 Behaviours of WSNs

3.1 Basic Behaviours

3.1.1 Composing a WSN Atom

WSN atoms are formed during the period of clustering. In this period, each sensor node is registered to a neighbouring BS.

3.1.2 Composing a WSN Molecule

Several WSN atoms (or sometimes only one WSN atom) form a WSN molecule. Like covalent bonds, in some occasions a WSN electron can work for more than one WSN atoms at the same time. Analogous to naming of chemical molecules, the WSN molecule is named after each concerned WSN atom. For example, the molecule demonstrated in Fig. 1 is named as $C^{11}B^{8}$.

3.1.3 Adding a WSN Electron

Whenever a new node is added into an atom, it will connect to its neighbours and decide its own level within the network. The addition of a new WSN electron will change the structure and the naming of the WSN atom. In Fig. 1, for example, as the boundary of the two WSN atom changes to the dashed line, the left atom will become C^{12}. If the new electron can only connect to the BS through some of the outmost electrons, then the addition will also change the number of layers.

3.1.4 Subtracting a WSN Electron

The more layers an atom has, the more active the external electrons. Outermost electrons may escape either at the attraction of another atom or by some other external force. Similarly, subtracting a WSN electron might occur when some WSN electrons escape away from their current WSN atoms to another less layered WSN atom as a result of some load balancing process or due to energy exhaustion. As shown in Fig. 1, when a WSN electron escapes away from the right WSN atom, the WSN atom becomes B^7.

3.2 Message Hopping

Within the AToM model, we consider a WSN photon as analogous to message exchange with an associated energy consumption. The energy consumption is approximately proportional to the length of the message. A WSN electron, once accepting a WSN photon, will turn to an excited state which persists until the electron hops the photon to a neighbour. By releasing the photon, the electron will return to a ground state. This process will iterate until the photon reaches the BS.

The whole routing action can be expressed as < H_1, H_2, H_3, …, H_N>, where each component hopping action H_I is carried out at node I. Suppose the environment at node I is e_I at some time point, the process is further denoted as a sequence of interleaved environment states E={ e_0 , e_1 ,… e_N } and actions $e_0 \xrightarrow{H_1} e_1 \xrightarrow{H_2} e_2 \xrightarrow{H_3} \dots \xrightarrow{H_{N-1}} e_N$.

See: E → Per

Act: I → Hop

Next: I × Per → I

Fig. 3. Coordinated routing functions

Because of resource limitations, particularly on communication ability, the bottleneck of routing is predominantly at the sensor nodes' side. The coordinated hopping behaviours of sensor nodes are listed in Fig. 3 as three major functions. A *see* function maps the environment states to perception; an action-selection function *act* maps from internal states to hopping actions; a *next* function maps internal state and perception to an new internal state. These functions are executed recursively until the message finally reaches the BS.

We adopt the simple assumption as described in [12], where the radio dissipates Ee = 50 nJ/bit to run the transmitter or receiver circuitry and Ea = 100 pJ/bit/m^2 to run the transmit amplifier. Therefore, to transmit an m-bit message a distance d expends

$$\begin{aligned} Et &= Ee \cdot m + Ea \cdot m \cdot d^2 \\ &= m \cdot (\, 50 + 0.1 \cdot d^2 \,)\ (nJ) \end{aligned} \tag{1}$$

To receive this message, the radio expends

$$\begin{aligned} Er &= Ee \cdot m \\ &= 50 \cdot m\ (nJ) \end{aligned} \tag{2}$$

If a message is sent through H hops and finally reaches the BS, then the hopping energy expense E_H is roughly expressed as

$$E_H = H \cdot Et + (\, H - 1 \,) \cdot Er \tag{3}$$

4 Management of Energy and Load Balancing

Having introduced the Atom naming conventions identified within Section 2 and the modelling of their behaviours presented within Section 3, we now turn our attention to the usage of such/ Specifically we now explore how this model could form the foundation of effective energy balancing amongst WSN electrons and load balancing among WSN subsets.

4.1 Load Balancing Between WSN Atom Levels

Assume that a WSN subnet contains one BS and N sensor nodes distributed into L levels, then the minimum energy consumption (hops at same levels are neglected) on each level for an operation of hopping a piece of a message from each node to the BS can be derived from Table 1.

Within a WSN, the restriction of sensor distribution on each hopping level is much looser than the distribution of electrons on different energy levels which is strictly defined by Pauli's law. However, the number of levels and the distribution of nodes on each level do make a difference to both energy efficiency and hopping cost within a given network. Energy balance is one of the major considerations in creating balance among levels, because it determines the life expectancy of the network. It can be seen from Table 1 that within a peer-to-peer network, sensor nodes closer to the BS have a heavier burden than those located at outer levels, if their hopping levels remain static. Consequently, the greater the number of nodes (within a fixed network) on low levels, the better the balance of the levels. Conversely the less the number of levels, the better the load balance of the network, assuming of course, no specific clustering techniques are applied. In addition, the fewer the number of levels the better the balance of power consumption among levels.

Table 1. Energy consumption of routing on each level for a round of data collection

Level	No. of nodes	Minimum Energy consumption on level
0 (BS)	1	External energy supply
1	N_1	$N\cdot Et + (N-N_1)\cdot Er$
2	N_2	$(N-N_1)\cdot Et + (N-N_1-N_2)\cdot Er$
3	N_3	$(N-N_1-N_2)\cdot Et + (N-N_1-N_2-N_3)\cdot Er$
…	…	…
K	N_k	$(N - \sum_{J=1}^{K-1}(N_J))\cdot Et + (N - \sum_{J=1}^{K}(N_J))\cdot Er$
…	…	…
L (Last)	N_L	$N_L\cdot Et$

4.2 Management of Load Balance Among WSN Atoms

We denote load balance level by an index called load imbalance level B_L, which is defined as the average unevenness of the corresponding numbers of WSN electrons of a group of N neighbouring WSN atoms {$atom_1$, $atom_2$, …, $atom_N$}.

$$B_L = (\sum_{i=1}^{N} | Load(atom_i)/Load_0 - 1 |)/N \tag{4}$$

Where $Load(atom_i)$ is a function to derive the number of WSN electrons of $atom_i$, $Load_0$ is the average number of WSN electrons of the group. The greater the B_L, the lower the load balance level.

```
// ATOMS: $atom_0$, $atom_1$
// Load(): get load of either $atom_0$ or $atom_1$
// Outmost(): get the outmost shared sensor node

If not_balanced
   Do while ($Load(atom_0) > Load(atom_1)+1$) or ($Load(atom_1) > Load(atom_0)+1$)
          Outmost(Smaller($Load(atom_0)$, $Load(atom_1)$))$\leftarrow$
                                        Outmost(Bigger($Load(atom_0)$, $Load(atom_1)$));
      Renew $atom_0$, $atom_1$;
   Enddo
Endif
```

Fig. 4. Load balancing algorithm between WSN atoms

Load balancing among WSN atoms, the process of lessening the unevenness of their electrons, is an important step toward prolonged lifetime and efficiency for the whole network. The balancing between two neighbouring WSN atoms could be achieved with the algorithm presented in Fig. 4, by moving one or more sensor nodes from a larger atom to a smaller atom. It is important to note that we assume that sensors within the network have a mobility capability and can either decide or be advised to migrate to a new destination. Such migration is achieved by aggregated hopping with sensors always binding to a given base station in a manner similar to electrons. Balancing of three or more WSN atoms can be achieved based upon repeated balancing between two neighbouring WSN atoms.

4.3 Management of Energy Balance Among Electrons on the Same Level

We denote energy balance level by energy imbalance level B_E, which is defined as the average unevenness of the corresponding energy level of a group of M neighbouring WSN electrons {electrons$_1$, electrons$_2$, …, electrons$_N$}.

$$B_E = (\sum_{i=1}^{M} | \mathrm{Energy}(\mathrm{Electron}_i) / \mathrm{Energy}_0 - 1|) / M \tag{5}$$

Where Energy(Electron$_i$) is a function to derive the energy level of electron$_i$, while Energy$_0$ is the average energy of the group {electrons$_1$, electrons$_2$, …, electrons$_M$}. The greater the B_E, the lower the energy balance level.

```
// Energy balancing among WSN electrons
//within the same WSN level

Do{
        swap ← 0;
        for (0<count < (No_electrons - 1); count++)
                {
                if (Energy(Electron[I] > Energy [I + 1])
                {
                        //swap nodes in hopping
                        priority(Electron[I])←→priority(Electron[I+1]);
                        swap ← 1;
                }
        }
} while (swap > 0);
```

Fig. 5. Algorithm for energy balancing among WSN electrons

In order to prolong life expectancy of a WSN network, energy balance among WSN same-level electrons should be embedded into the management software. Many routing protocols support energy balance [6] [8] [9] [10] [12]. Among them, LEACH [12] is a clustering-based protocol that utilizes randomized rotation of cluster-heads to evenly distribute the energy load among the sensor nodes in the network. Within the management software, such rotation manifests itself as a swapping operation. Fig. 5 demonstrates the algorithm we adopt for energy balancing among electrons on the same level.

5 Case Studies

A simulation was carried out on a randomly distributed sensor networks, where the above-mentioned balancing management was implemented, as demonstrated in the following case studies.

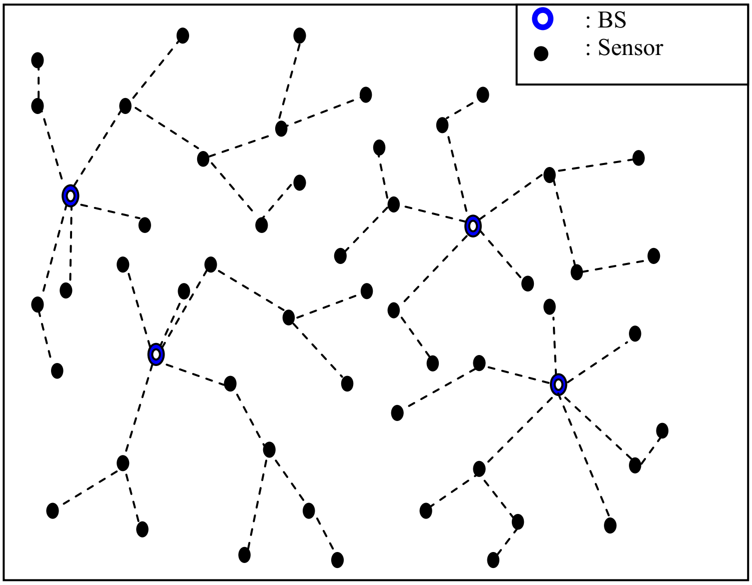


Fig. 6. Clustering and balancing of WSN atoms

The first case study deals with the clustering of a WSN consisting of 4 WSN nucleus and 51 WSN electrons where the threshold of the load imbalance level is set to 20%. These sensor nodes were clustered dynamically according to their neighbours. The execution of the clustering algorithm results in a balanced structure of WSN atoms, D^{14}, C^{12}, D^{14}, and C^{11}, as a topological compromise shown in Fig. 6. From Eq.(4), the load imbalance level B_L of this specific case is 12%. This represents an 8% improvement with a reduced load imbalance level (see Section 4.2).

A second case study dealt with the energy balance of a WSN atom, C^{11}, over a period of time. As demonstrated in Fig. 7, the white curve is the initial energy distribution of the 11 WSN electrons, while the black one is the energy distribution of these nodes some time later. It can be calculated from Eq. (5) that their initial energy distribution {0.98, 0.78, 0.71, 0.78, 0.6, 0.68, 0.74, 0.91, 0.78, 0.66, 0.82} has an imbalance level of about 10.5%. The energy-balancing algorithm improves the energy distribution to the black curve {0.59, 54, 0.5, 0.53, 0.45, 0.48, 0.51, 0.56, 0.52, 0.5, 0.54} which has an energy imbalance level of about 5.6%. The results demonstrate that through the execution of the energy-balancing algorithm, the energy imbalance level among these WSN electrons is improved by 4.9%.

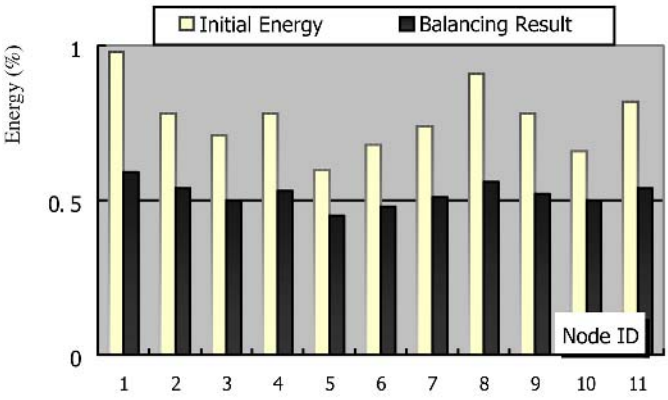


Fig. 7. Schematic view of energy balance

6 Conclusions

The Atomic Topology Management introduced in this paper is novel in its adoption of chemical structures and chemical reactions as a metaphor for WSN management. This is the first work that has advocated drawing such parallels as an instrument to intelligent management of WSNs and thus imbuing autonomic properties to such networks. This approach can be applied to the design and management of sensor networks on the MAC layer and routing layer. More specifically, it helps in evaluating WSNs' energy efficiency of routing, scalability, network lifetime, coverage and load balancing. The algorithms described within this paper have already been utilized within mote-based WSNs.

On-going and future research will focus on the completion of the management platform. In particular we are further extending the chemical analogy by investigating how to characterise not individual base station and sensor subnets, but rather the aggregation of numerous such subnets. We examine issues of overall stability of the *compound/network* and consider how this effects or inhibits reactions. We examine the *Subnet Periodic Table* as a classification mechanism for subnet structures. This borrows heavily from its chemical counterpart. We are exploring the concept of *groups* and *periods* and other classifications. Additional work considers subnet composition theories that will obviously stand in contrast to the Aufbau principle. Our research also examines such issues as chemical reactions caused between subnet components together with the possibility of co-valence bonds and how these could facilitate network adaptivity.

While Atomic Topology Management of Wireless Sensor Networks is a work in progress the efficacy of the model has already been demonstrated in effective naming and energy and load balancing algorithms. While these are rudimentary at this stage they demonstrate how the model can underpin autonomic WSN management and adaptivity delivering significant performance improvements.

Acknowledgements

S. Shen, G.M.P. O'Hare, D. Marsh, D. Diamond, and D. O'Kane gratefully acknowledge the support of Science Foundation Ireland under Grant No. 03/IN.3/1361.

References

1. J. Hill, R. Szewcyzk, A. Woo, S. Hollar, D. Culler, K. Pister, "System Architecture Diretions for Networked Sensors", Proceedings of Ninth International Conference on Architectural Support for Programming Languages and Operating Systems, November 2000.
2. I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci. "Wireless Sensor Networks: A Survey", Computer Networks, 38(4): 393-422, March 2002.
3. H. Dai, R. Han, "A Node-Centric Load Balancing Algorithm for Wireless Sensor Networks", IEEE GLOBECOM - Wireless Communications, 2003
4. P. H. Hsiao, A. Hwang, H. T. Kung, and D. Vlah. "Load-Balancing Routing for Wireless Access Networks", IEEE Infocom, April 2001.
5. Q. Li, J. Aslam, and D. Rus. "Hierarchical power-aware routing in sensor networks". In Proceedings of the DIMACS Workshop on Pervasive Networking, May 2001.
6. A. Manjeshwar and D. P. Agrawal. "Teen: A routing protocol for enhanced efficiency in wireless sensor networks". In 1st International Workshop on Parallel and Distributed Computing Issues in Wireless, 2001.
7. J. Zhao, R. Govindan and D. Estrin, "Residual Energy Scans for Monitoring Wireless Sensor Networks", IEEE Wireless Communications and Networking Conference (WCNC'02), Orange County Convention Center, Orlando, FL, USA, 17-21 March, 2002.
8. C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", In the Sixth Annual International Conference on Mobile Computing and Networking, Boston, MA, USA, August 2000.
9. Y. Yu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks", Technical Report UCLACSDTR-01-0023, Computer Science Department, University of California at Los Angeles, May 2001.
10. A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges for reliable multi-hop routing in sensor networks", SenSys'03, Log Angeles, CA, USA, November 2003.
11. X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks", First ACM Conference on Embedded Networked Sensor Systems (SenSys'03), Los Angeles, CA, November 2003.
12. W. Rabiner, Heinzelman, A. Chandrakasan and Hari Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Micro-sensor Networks", In Proceedings of the 33rd International Conference on System Sciences (HICSS '00), 2000.
13. J. Pan etc, Topology Control for Wireless Sensor Network, International Conference on Mobile Computing and Networking, Proceedings of the 9th annual international conference on Mobile computing and networking, San Diego, CA, USA, PP. 286 – 299, 2003.
14. A. Salhieh, J. Weinmann, M. Kochhal, and L. Schwiebert, Power Efficient Topologies for Wireless Sensor Networks, in International Conference on Parallel Processing, pages 156--163, Sept. 2001.

An Architecture for Autonomic Management of Ambient Networks

Marcos A. Siqueira[1,2], Fabio L. Verdi[2], Rafael Pasquini[2],
and Mauricio F. Magalhães[2]

[1] CPqD Telecommunications R&D Center,
Rod. Campinas - Mogi-Mirim, km 118,5, 13086-902 Campinas, SP Brazil
[2] State University of Campinas, UNICAMP, FEEC, DCA,
P.O. Box 6101, 13083-970 Campinas, SP Brazil
{siqueira, verdi, pasquini, mauricio}@dca.fee.unicamp.br

Abstract. This paper proposes an architecture for Ambient Networks (AN) management, which is based on components such as Autonomic Networking and Policy Based Network Management. The main goal is to allow AN management aligned to the network requirements, incorporating functionalities such as self-configuration, self-management, self-healing and self-protection. In the proposed architecture, all these functionalities are controlled by a distributed policy-based system, allowing network configuration based on business policies.

1 Introduction

With the advent of wireless technologies such as IEEE 802.11, 802.16, cellular 3G, 4G (all IP), the connection of users to networks and even the connections among different networks tend to be dynamic. An example of such scenario, presented in the context of the Ambient Networks (AN) project [AN06], is a train of passengers operating an internal WLAN (Wireless Local Area Network), communicating to the external world via wireless access networks available through the railway. The train might have different kinds of wireless interfaces, connecting to the better network available at each moment, maintaining the passengers connections. Therefore, the train has to negotiate parameters such as the connection conditions, even without having a previous knowledge of the resources provided by other networks.

These kinds of networks become very difficult to be operated by humans without the use of automated management tools, due to the quantity of protocols and logical entities that must be configured, the amount of interconnections and its dynamicity, the great quantity and widespread of network nodes, besides the heterogeneity of technologies and types of equipments from many vendors. This heterogeneity generates greater complexity in network operation, resulting in increase of OPEX (OPerational EXpenditures) since specialized engineering, operation and support teams are required for each solution employed.

It has been argued by the academy and some standardization bodies that network management automatization is the solution to: operation cost reduction, resources optimization, security control, and mainly the responsiveness in

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 255–267, 2006.

troubleshooting. Besides, the application of the Autonomic Networking [Str05] concept hides the network complexity, by means of self-management, self-optimization, self-defense and self-configuration, in a manner aligned to the end business. For complete automatization of operation tasks, network management tools might act as integrated parts of the enterprise process model. As an example, the CRM (Customer Relationship Management) might be integrated to the services activation tools, that should be integrated to the NMS (Network Management System) allowing automatic services provisioning according to the CRM commands generated by client requests or others.

This paper discusses how the autonomic networking can help in the solution of the problems presented in the previous paragraphs, through the automatization and consequent simplification of network operation. The proposal is that the concept of network policies should be used to regulate the autonomic network operation. Specifically, for validating the proposed approach, it is proposed a distributed architecture that supports autonomic management and operation of Ambient Networks. For validating the proposal, an infra-structure of distributed policy agents is implemented using *web services*, which are very suitable for overlay networks, allowing automatic discovery of resources and open communication, both required by Ambient Networks.

The rest of this paper is organized as follows: Section 2 presents the concept of Ambient Networks and its requirements, Section 3 presents the concept of Autonomic Networks, Section 4 presents a brief about Policy-Based Network Management, Section 5 presents an architecture for autonomic management of ambient networks, describing its functional elements and the strategies for the system implementation in a distributed way, finally Section 6 presents the conclusion and future works.

2 Ambient Networks

Ambient Networks (AN) [AN06] is a project aiming at investigating future communication systems. One of the main goals is to allow dynamic composition of heterogeneous networks from different operators and/or different technological domains transparently to the users and services.

Nowadays, Internet Protocol is the *lingua franca* to enable information exchange through networks. Nevertheless, there is divergence in the network control layer, since different mechanisms are needed, for instance, to implement VPNs, security, QoS, Multicast, and others. This diversity of control mechanisms is a barrier to reach the level of integration desired in the Ambient Networks. Therefore, ANs present as challenge, the definition of universal essential control functions aiming at reaching a scenario that allows dynamic network composition and enhanced mobility control. These functions might take into account heterogeneous environments with different network techonologies and services as well as varied network management functions. A conceptual framework, formed by three basic principles was defined by [SEPP05]. These principles are:

1. **ANs must be built over open platforms:** the goal is to eliminate architectural constraints about what or who might connect to what or who, enabling network services not only for nodes, but also to whole networks. This strategy surpasses the limitations inherent to node centric architectures, mainly in scenarios with PANs (Personal Area Networks), mobile networks, ad hoc networks, connected to one another.
2. **ANs are based on self-composition and self-management:** the composition of networks so that packets could be forwarded would be simple, but the composition of networks in a managed way, maintaining advanced functions such as QoS, security and mobility is a complex task. The goal is ANs to have self-composition and self-management functions as basic premise.
3. **AN Functions can be added to existent networks:** the philosophy of ANs is not to create a totally new network, but aggregate the wished functionalities to current networks, allowing them to integrate not only at the packet forwarding level, but also at the control plane, enabling the operation of services end-to-end through the cooperation of the different networks.

The main concept defined by the AN project is the ACS (Ambient Control Space), which manages the functionalities related to control and data transport through a set of interfaces for services and applications. The ACS is composed by a number of interdependent complex control functions that are being developed in the context of the AN project. Examples of these functions are the support to multi-radio accesses, mobility, security, management and smart media routing context management.

An Ambient Network has well defined control interfaces to other ANs, service platforms and applications. Aiming at allowing cooperation between different ANs, it is defined the ANI (Ambient Network Interface), which allows the connection of the ACS functions of a given AN to the ACS of connected ANs. On the other hand, the access to the services of a given AN is performed via ASI (Ambient Services Interface). Together, the ANI and ASI might support: plug and play connection and composition of ANs, network reconfigurations, support to mobile networks and a single interface to external entities, even in the case of cooperation between two or more ANs. Figure 1 shows the architecture defined for Ambient Networks, including the Ambient Control Space, ANI and ASI interfaces.

2.1 Ambient Network Management

Ambient Networks require management systems able to act in a dynamic and automatic way, allowing the networks to perform tasks such as auto-composition without the need of human intervention. To be able to perform these tasks, AN management systems might be dynamic and auto-managed, acting actively in the network. Other important requirement presented by [NGA$^+$04] is the ability to connect the management systems of two ANs consistently in the occurrence of AN composition, or even in the event of separation of these systems in a predictable and consistent way. Among the techniques already proposed for AN management, the following are worth listing:

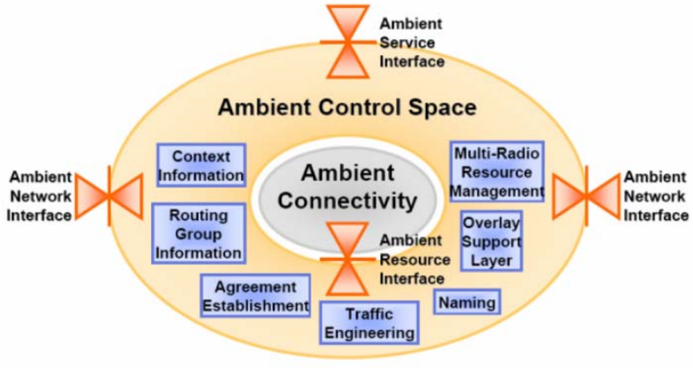

Ambient Service Interface
Ambient Control Space
Context Information
Ambient Connectivity
Multi-Radio Resource Management
Ambient Network Interface
Ambient Network Interface
Routing Group Information
Overlay Support Layer
Ambient Resource Interface
Agreement Establishment
Naming
Traffic Engineering

Fig. 1. Ambient Network Architecture

1. **Point-to-point model:** it is a distributed management model in which the main characteristic is the possibility of interaction among peer elements or networks, allowing for instance, dynamic composition and decomposition of networks.
2. **Pattern-based model:** the paradigm of pattern-based management deals with algorithms for processing and aggregation of management information in a distributed system. A key feature is the separation of the tasks semantics from the management operations (management control flow).
3. **Plug and play model:** the main objective of this model is to enable automatic configuration of new network elements, allowing them to join an AN with support to the domain features. Other strategies include application of traffic engineering for resources optimization in the AN, besides integration of management strategies.

The next section presents Autonomic Networking scheme as a possible architecture for distributed, policy-based management of Ambient Networks.

3 Autonomic Networking

When the idea of Autonomic Computing (AC), introduced by IBM in 2001 [KC03] is applied at the level of computer networks, the result is the concept of Autonomic Networking. The application at the network level of the autonomic systems requirements such as self-management, self-configuration, self-optimization, self-healing and self-protection enables drastic simplification and automatization of the network operation. Therefore, is perceived that the concept of Autonomic Networking is a superset of the Policy-Based Network Management (PBNM) concept. While PBNM is focused in network auto-configuration based on certain conditions, the concept of Autonomic Networking includes broader functions such as self-protection, self-optimization and self-healing. Following, some possible solutions for the implementation of the features needed for the construction of an autonomic network are discussed:

1. **Self-management:** it is a desirable feature, but it is not commonly available in current network management tools. Self-management plays a very important role due to the complexity and distributed nature of autonomic systems. Self-management of autonomic systems is an issue to be more deeply investigated by the academy.
2. **Self-configuration:** a possible way to perform self-configuration in networks is through the application of the PBNM concept, in the sense that network events and conditions can be determinant to trigger automatic reconfiguration of elements, ruled by pre-configured policies. The following references discuss about the auto-configuration of autonomic systems using policies: [KY03] proposes a language named JSPoon aiming at integrating autonomic functions in autonomic agents which might be installed at the systems to be managed; [BTBR04] proposes an autonomic middleware for control of the service used to orchestrate distributed and self-healing applications; and [BGJ$^+$04] proposes a policy-based framework for managing autonomic database systems based on business goals, allowing automatic data management based on events.
3. **Self-optimization:** network optimization is an issue in continuous research by the academy. The most relevant initiative is the deployment of traffic engineering, facilitated by MPLS (Multiprotocol Label Switching) [DOA02]. Notwithstanding, the network optimization can be reached through traffic control and congestion prevention. Specifically, [Aib04] presents a new paradigm focused in self-optimization according to high level business goals, such as profit maximization. This paradigm replaces traditional optimization schemes, based on metrics related to IP as resources availability. This work proposes an autonomic process responsible for such optimizations, and a set of logical entities and its communication interfaces to implement this process.
4. **Self-healing:** it is evaluated that self-healing is the more difficult feature to be implemented in the point of view of a computer system. Systems can be designed with some degree of redundancy being, therefore, fault tolerant. Nevertheless, self-healing of failed components might not be performed automatically without human intervention.

   Currently, mechanisms used for fault recovery in networks are very developed. In the core layer, [LRP05] presents a functional description of the extensions needed by GMPLS (Generalized Multi-Protocol Label Switching) for adding the functionalities of protection and restoration. The IETF working group BFD (Bidirectional Forwarding Detection) is in period of specification of a protocol (named BFD) for fast detection of failures in links. RFC4090 [PSA05] specifies a mechanism for fast reroute of LSPs (Label Switched Paths) within MPLS networks. In the distribution layer, metro network technologies also support several mechanisms for fault tolerance, comprising: RPR (Resilient Packet Ring) [RPR06], RSTP (Rapid Spanning Tree Protocol) [RST] and EAPS (Ethernet Automatic Protection Switching) [SY03]. In the access layer, the following mechanisms are suitable for fault tolerance: RF diversity, dial backup with DDR (Dial on Demand Routing), access link redundancy, CPE router duplication, protocols for redundancy such as VRRP (Virtual Router Redundancy Protocol) [KWW+98] and GLBP (Gateway Load Balancing Protocol) [GLB06].

5. **Self-protection:** the vendor Cisco Systems launched recently a bundle named "Cisco Self-Defending Network" [SD06]. The promise is that the implementation of this strategy allows the network to identify, prevent and adapt to threats. The first phase of the project includes integration of security features to network elements including routers, switches, wireless access points, and others network appliances. The second phase includes a mechanism named NAC (Network Admission Control) which allows network elements enabled for security to intercommunicate in a collaborative way, allowing security functions to be extended to equipments of individuals which eventually connect to other networks bringing risk to the corporate network.

   Adaptive defense from threats is the last phase, which helps to minimize security risks, dealing with threats dynamically in multiple layers, allowing a more strict control of traffic, endpoints, users and applications. This last strategy aims at protecting every packet and flow that pass throughout the network. Therefore, it is noted that the initiative Self-Defending Network contributes to the self-protection feature wished for autonomic networks.

4 Policy Based Network Management

In the last years, it has been developed architectures, information models and policy protocols aiming at reaching the goals of PBNM systems, such as mapping high level business rules into network configurations, according to the occurrence of a set of events, and even the analysis of applicable conditions.

Study areas of PBNM systems can be classified into: policy languages, policy information models, PBNM architectures, conflict detection and resolution strategies, policy protocols and policy edition strategies. Many works related to PBNM have been published. Works related to policy languages include [LSDD00], [AMN02] and [LK03]; works related to policy information models include [Str02], [MESW01] and [SMFdC04], works related to conflict detection and resolution include [LS99] and [Dun02]; works related to PBNM architectures include [SMC02] and [SNRLM05]. The references listed above are a small sample of the number of academic works related to PBNM systems. As a result to a simple search for papers at the IEEE repository, more than 50 papers related to application of PBNM in specific technologies or scenarios can be found.

From the commercial systems perspective, vendors of network management tols such as Hewlett-Packard, Micromuse, Extreme Networks, Cisco Systems, have launched their own PBNM modules, coupled to the NMS systems, mainly for specific goals such as policy-based QoS and security management.

The next section presents a strategy for managing and controlling Ambient Networks, employing Autonomic Networking and PNNM strategies.

5 Autonomic Management of Ambient Networks

This work proposes an approach for Ambient Networks management, applying Autonomic Networking concepts aiming at implementing support for mobility, network auto-composition, distributed management and other control functionalities. In order to develop the system architecture, the following premises are assumed: (1) The operation scenario will be a set of Ambient Networks, part of them nested, part of them intercommunicating through ANI and ASI interfaces. These ANs might compose or decompose with other ANs at any moment, and also might accept the connection of new nodes with full support to auto-configuration; (2) The network devices might support a common GLL (Generic Link Layer) in order to allow connectivity at the link layer level.

The architecture proposed has the following main features: (1) The ACS has an Autonomic Management Layer (AML) which might be distributed through one or more ANs. The control messages are exchanged throughout the ANI, allowing seamless connection of two or more ANs in the management point of view; (2) The AML might be implemented totally distributed through agents installed at the network elements, providing support to communication with other ANs, allowing establishment of AN control functions; (3) The AML might operate automatically, without the need of human intervention in runtime, nevertheless being possible to perform tuning in the configurations that affect the behavior of the AN; (4) The operation of the AML might be policy-based, preferably being ruled by high level business policies.

Figure 2 presents a scenario composed by four ANs, in which AN1, AN2 and AN3 are composed, interconnected by ANI and ASI interfaces, while AN4 is in composition with AN2. According to the figure, each AN might have or not a management element named APS (Autonomic Policy Server). In the example shown, only AN2 doesn't have a APS. Each existent APS can instantiate APAs (Autonomic Policy Agents) and prosecute some control over them in the scope of one AN or other connected AN, if the last doesn't have its own APS. In this scenario, the proposed architecture requires the occurrence of information exchange among APAs in the same AN and among APAs of different ANs through the ANI interface. As far as AN2 doesn't have a APS, the agents APA2a, APA2b and APA2c are instantiated and partially controlled by the APSs of the connected ANs.

Comparing the presented network and management architecture with common PBNM architectures, is noted that the function of PDP (Policy Decision Point) is performed in a distributed way by the Autonomic Policy Servers and by the Autonomic Policy Agents. On the other hand, the PEP (Policy Enforcement Point) functions are performed only by the Autonomic Policy Agents. The following subsections detail the functions of the APS and APA.

5.1 Autonomic Policy Agent

The APAs might be instantiated manually by the network operator or by the APS, allowing the AN to manage its resources in a distributed and autonomic

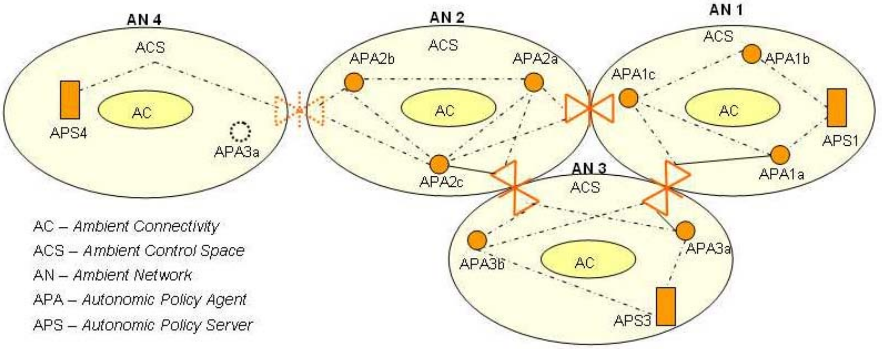


Fig. 2. Ambient Network Distributed Management

fashion. The APAs of a given domain will always establish logical communication channels for exchanging control messages. The logical paths might be configured with a fault tolerant strategy, as seen in Figure 2, where in the AN1 network, each APA has at least two logical routes to every other APA or APS in the domain, or to other ANs connected via ANI.

Each APA is responsible for performing the following tasks: (1) Auto-configuration of network elements by which the given APA is responsible; (2) Take local policy decisions about configurations, and adjusts to be performed at the network elements, mostly related to QoS and mobility control; (3) Participate actively in authentication and authorization of new nodes and new ANs trying to join the network, enhancing scalability and security of the whole system; (4) Keep neighbor relationship and logical control paths to other APAs and the APS; and (5) Have the ability to maintain operation even in the occurrence of failure in communication with the APS.

5.2 Autonomic Policy Server

The APS is responsible for: (1) Keep a centralized copy of the whole policy repository of its respective domain of control, formed by one or more ANs; (2) Instantiate or activate APAs in a given AN aiming at optimizing the management functions of this AN; (3) Receive requests from APAs for decisions about complex situations or conditions, which require a broader vision than is available at the APA level; and (4) Provide an interface to the network administrator for entering policies at the whole system.

5.3 Policy Construction

In the context of this project, it is designed an advanced mechanism for interpretation and execution of policies. The policy engine allows for complex policy structure, modeled not only by traditional ECA (Event-Condition-Action), but also considering the concept of situation, introduced by [AE02]. The policy engine designed for the PDP is shown in Figure 3.

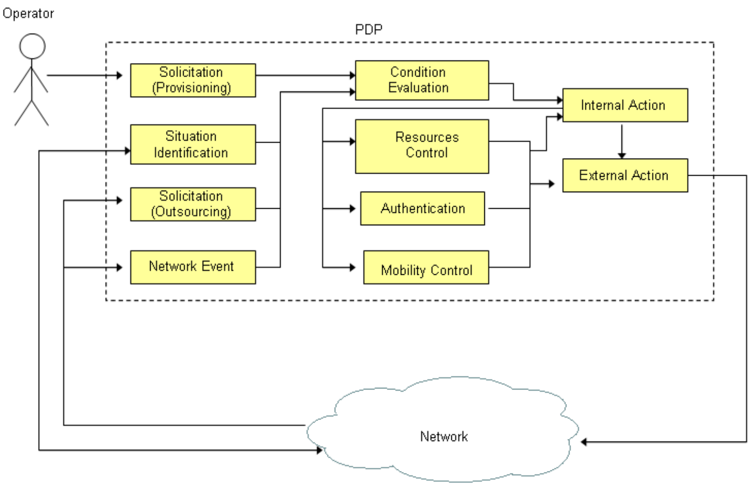


Fig. 3. Policy evaluation and execution flow

The PDP is formed by three main stages. The first one is responsible for receiving solicitations, events or for identifying a situation. The second stage is responsible for analyzing conditions, authentication, analysis of parameters related to mobility control, resources management (eg.: available bandwidth). The third stage is responsible for performing internal and external actions. Internal actions are executed when the analysis of further conditions is needed and external actions consist of performing specific configurations in the network elements.

5.4 Distributed Policy Engine Implementation

Figure 3 has shown the policy engine in a centralized fashion. Nevertheless, the AN architecture requires a distributed management system that suits its requirements. Therefore, the components of the policy engine were designed as independent Policy Agents, communicating to each other through a *web services* infrastructure. The use of *web services* as integration technology between different applications facilitates the development of future systems. It has been argued that distributed technologies such as CORBA, DCOM, RMI and others already exists and could be employed, but all of them adopt distribution mechanisms strongly coupled, synchronous communication and weak interoperability among different technologies. On the other hand, using XML and HTTP, *web services* tend to be weakly coupled. Besides, *web services* facilitate the development of business solutions among enterprises due their ability for publishing enterprise solutions as services available throughout the network.

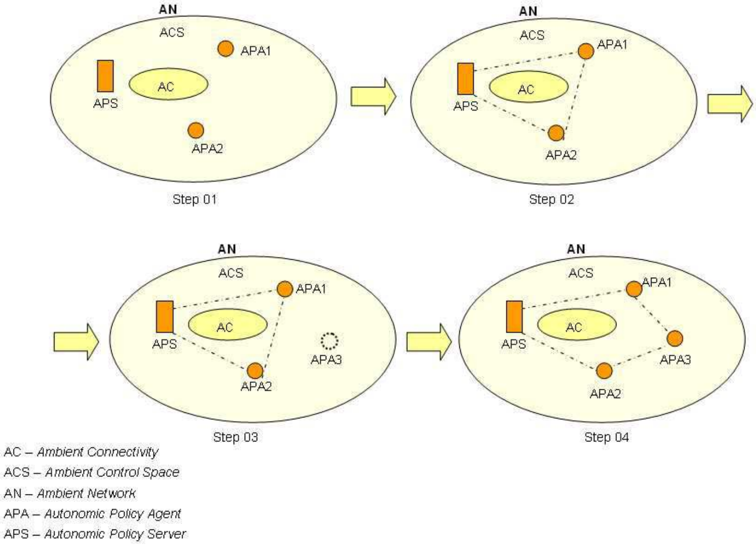


Fig. 4. Sample of the policy flow operation

The distributed implementation was designed using the concept of policy flow. The APS is responsible for throwing the policy activation. According to Figure 04, the first step performed by the APS is determining the set o APA that will take part of the ring, creating the logical paths between them, then in step 02 the policy is distributed throughout these agents and activated. From this moment on any of these agents is responsible for triggering policy analysis and execution in the occurrence of a given event, solicitation, or identification of a situation. The policy flow is comprised by each agent running a set of methods of the policy and sending the policy to the next agent in the policy ring. Each agent receives a policy in a given state of the evaluation, indicating the next method to be executed. The local agent performs the tasks related to that method and dispatches the executing policy to the next agent, according to the decisions made. If a given agent doesn't know for any reason where to send the policy, it appeals to the APS which has a global view of the policy flow, and is able to perform the needed tasks of evaluating and executing the policy. The APS is published as a *web service* so that the PAs can find its available resources.

In the step 03 shown in Figure 4 a new Policy Agent appears. In this case, a situation is identified by the policy configured at the Policy Agent APA1, which carries out the process of policy evaluation, performing condition evaluation, which results in the execution of an internal action. The internal action delegates the authentication of the new APA to the APS. Then, the APS performs step 04 shown, authenticating APA3, and reconfiguring the policy ring with the new agent. The example presented is very simple, but the policy engine shown in Figure 4 allows the deployment of complex policies by the system.

6 Conclusion and Future Works

This paper presented a proposal of architecture for Ambient Networks management. As a new approach, Autonomic Networking and Policy-Based Network Management concepts are employed forming the base of the proposed architecture. The paper presented the requirements and concepts of Ambient Networks, discussing about propositions available at the literature and available tools that can fit these requirements. Finally, the implementation architecture of the distributed autonomic policy-based management system is described.

The work described in this paper is part of the author's Doctorate thesis [Siq06]. As future works, is proposed to be explored the strategies for APS discovery, algorithms for APA instantiation, strategies for creation and maintenance of logical paths between PAs in a domain and inter-domains, and the refinement of the policy flow. Finally, the system will be validated over a real testbed AN network.

References

[AE02] A. Adi and O. Etzion. The Situation Manager Rule Language. In *Proceedings International Workshop on Rule Markup Languages for Business*, 2002.

[Aib04] S. Aiber. Autonomic Self-Optimization According to Business Objectives. In *Proceedings of the International Conference on Autonomic Computing (ICAC04)*. IEEE, 2004.

[AMN02] X. Ao, N. Minsky, and T. D. Nguyen. A Hierarchical Policy Specification Language and Enforcement Mechanism, for Governing Digital Enterprises. In *Proc. of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, June 2002.

[AN06] AN. Project: EU FP6 Program Ambient Networks, April 2006.

[BGJ+04] M. Bhide, A. Gupta, M. Joshi, M. Mohania, and S. Raman. Policy Framework for Autonomic Data Management. In *Proceedings of the International Conference on Autonomic Computing (ICAC'04)*. IEEE, 2004.

[BTBR04] N. Badr, A. Taleb-Bendiab, and D. Reilly. Policy-Based Autonomic Control Service. In *Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY04)*. IEEE, 2004.

[DOA02] B. Jabbari D. O. Awduche. Internet Traffic Engineering Using Multiprotocol Label Switching (MPLS). *Journal of Computer Networks (Elsevier Science)*, 40(1), September 2002.

[Dun02] N. Dunlop. *Dynamic Policy-Based Management in Open Distributed Environments.* PhD thesis, Department of Computer Science and Electrical Engineering of the University of Queensland, September 2002.

[GLB06] GLBP. Gateway Load Balancing Protocol. http://www.cisco.com search GLBP, 2006.

[KC03] J. O. Kephart and D. M. Chess. The Vision of Autonomic Computing. *IEEE Computer Magazine*, 36:41–50, January 2003. ISSN:0018-9162, Issue 1.

[KWW⁺98] S. Knight, D. Weaver, D. Whipple, R. Hinden, D. Mitzel, P. Hunt, P. Higginson, M. Shand, and A. Lindem. Virtual Router Redundancy Protocol. IETF RFC2338, April 1998.

[KY03] A. V. Konstantinou and Y. Yemini. Programming Systems for Autonomy. In *Proceedings of the Autonomic Computing Workshop Fifth Annual International Workshop on Active Middleware Services*, ISBN 0-7695-1983-0/01. IEEE, 2003.

[LK03] A. Joshi L. Kagal, T. Finin. A Policy Language for a Pervasive Computing Environment. In *Fourth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY03)*, 2003.

[LRP05] J. P. Lang, B. Rajagopalan, and D. Papadimitriou. Generalized Multi-Protocol Label Switching (GMPLS) Recovery Functional Specification. nternet Draft draft-ietf-ccamp-gmpls-recovery-functional-04.txt, April 2005.

[LS99] E. C. Lupu and M. Sloman. Conflicts in Policy-Based Distributed Systems Management. *IEEE transactions on software engineering*, 25(6), November 1999.

[LSDD00] E. Lupu, M. Sloman, N. Dulay, and N. Damianou. Ponder: Realising Enterprise Viewpoint Concepts. In *In Proceedings 4th Int. Enterprise Distributed Object Computing*, 2000.

[MESW01] E. Moore, E. Ellesson, J. Strassner, and A. Westerinen. Policy Core Information Model – Version 1 Specification. IETF RFC 3060, February 2001.

[NGA⁺04] J. Nielsen, A. Galis, H. Abrahamsson, B. Ahlgren, M. Brunner, L. Cheng, J. A. Colas, S. Csaba, A. Gonzalez, A. Gunnar, G. Molnar, and R. Szabo. Management Architectures and Approaches for Ambient Networks. In *12th WWRF meeting*, Toronto, Canada, November 2004. NLE-PR-2004-65.

[PSA05] P. Pan, G. Swallow, and A. Atlas. Fast Reroute Extensions to RSVP-TE for LSP Tunnels. IETF RFC4090, May 2005.

[RPR06] RPR. Resilient Packet Ring Alliance. http://www.rpralliance.org, 2006.

[RST] RSTP. IEEE Standard - 802.1w - Rapid Reconfiguration of Spanning Tree, supplement to ISO/IEC 15802-3:1998.

[SD06] Self-Defending. Cisco Self-Defending Network. http://www.cisco.com search self-defending networks, 2006.

[SEPP05] Andreas Schieder, L. Eggert, N. Papadoglou, and F. Pittmann. Components and Concepts of the Ambient Networks Architecture. Wireless World Research Forum WWRF, 2005.

[Siq06] Marcos A. Siqueira. *Policy-based Autonomic Management (work in progress)*. PhD thesis, Unicamp, 2006.

[SMC02] M. A. Siqueira, M. F. Magalhães, and E. Cardozo. A Policy Management Architecture for MPLS Networks. In *Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems (PDCS 2002)*, 2002.

[SMFdC04] M. A. Siqueira, M. F. Magalhães, L. J. L. Farias, and M. C. de Castro. A BGP/MPLS PPVPN Management Information Model and a J2EE-based Implementation Architecture for Policy and Web-Based Configuration Management Systems. In *Proceedings of IEEE ICN'04 - 3rd International Conference on Networking*, 2004.

[SNRLM05] M. A. Siqueira, N. A. Nassif, R. A. Resende, and M. Lima-Marques. Policy-Based Architecture for QoS Management in Enterprise IP Networks. In *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management*, 2005.

[Str02] J. Strassner. DEN-ng: Achieving Business-Driven Network Management. In *NOMS*, 2002.

[Str05] J. Strassner. Autonomic Networking - Theory and Practice. Tutorial 4, IM2005, Nice-France, May 2005.

[SY03] S. Shah and M. Yip. Extreme Networks' Ethernet Automatic Protection Switching (EAPS) Version 1. IETF RFC 3619, October 2003.

Autonomic Communications: Exploiting Advanced and Game Theoretical Techniques for RAT Selection and Protocol Reconfiguration

Eleni Patouni[1], Sophie Gault[2], Markus Muck[2], Nancy Alonistioti[1],
and Konstantina Kominaki[1]

[1] Communication Networks Laboratory, Department of Informatics & Telecommunications,
University of Athens, Athens, Greece
{elenip, nancy, kominaki}@di.uoa.gr
[2] Motorola Labs, Paris, France
{sophie.gault, markus.muck}@motorola.com

Abstract. The Autonomic Communications concept emerges as one of the most promising solutions for future heterogeneous systems networking. This notion implies the introduction of advanced mechanisms for autonomic decision making and self-configuration. To this end, this paper proposes an integrated framework that facilitates autonomic features to capture the needs for RAT selection and device reconfiguration in a Composite Radio Environment. Specifically, a game theoretical approach targeted to the definition of appropriate policies for distributed equipment elements is presented. Thus, the user terminals are able to exploit context information in order to i) identify an optimum trade-off for (multiple) Radio Access Technology (RAT) selection and ii) adapt the protocol stack and respective protocol functionality using a proposed component based framework for transparent protocol component replacement. Simulation and performance results finally show that the proposed mechanisms lead to efficient resource management, minimizing the complexity on the network and terminal side as well as keeping the required signaling overhead as low as possible.

Keywords: autonomic networking, cognitive networks, reconfiguration.

1 Introduction

Future beyond 3rd Generation (B3G) systems are expected to exploit the full benefits of the diversity within the radio eco-space, composed of wide range of systems such as cellular, fixed, wireless local area and broadcast. In this framework, it is important to provide suitable means on the network and terminal side serving as an enabler for this vision. Such vision is captured by the notion of autonomic communications which provides the grounds for the deployment of advanced concepts, including a device agnostic and protocol independent approach for an hierarchy of systems with self-managing, self-configuring and self-governance features. Beyond the conceptual merits of such an approach, the following key issues need to be addressed in a practical context: i) Manage the complexity on the network and user terminal side and

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 268 – 284, 2006.

provide policy communication means, ii) Minimize required signaling overhead and iii) Provide means for the device dynamic adaptation following the decision for RAT selection.

The first of these items typically motivates a distributed system concept as analyzed in [1] in the context of autonomic communications where self-managing devices with behavior controlled by policies are introduced. Furthermore, we assume the introduction of a suitable cognitive channel which covers, besides policy related information, future context data helping the devices to perform decisions. Item ii) relates to the policies themselves, leading to the observation that *simple, global policies* (applicable to all users) should be preferred to user specific rules in order to assure a minimum signaling overhead. In addition item iii) is related to the introduction of a framework incorporating the necessary mechanisms that enable the dynamic adaptation/reconfiguration of the protocol stack.

In the context of this paper, all these principles are highlighted; the rest of this contribution is organized as follows: Section 2 defines the general study framework portraying the problem that is examined in this contribution. A RAT selection analysis for a simple two-user context, based on game theoretic tools is presented in section 3. A framework for the dynamic protocol reconfiguration over heterogeneous RATs is analyzed in section 4. Finally, related work and conclusion remarks as well as directions for future research are highlighted in section 5 and 6 respectively.

2 Problem Statement

In this analysis, a composite radio environment, in terms of a distributed network of heterogeneous Radio Access Technologies (RATs), is considered, as illustrated below (Fig. 1). A multitude of users is assumed to compete for access to one or several RATs and one or several distinct communication channels (in terms of spectrum usage) in parallel. An efficient operation requires suitable RAT/channel selection algorithms: in heterogeneous and reconfigurable wireless systems, terminals and network equipments should incorporate enhanced capabilities for adapting to the drastically changing environment.

Towards this direction, this paper analyzes an integrated framework that facilitates autonomic features to capture the needs for RAT selection and device reconfiguration in a Composite Radio Environment. At first, the process of selecting a RAT targeted to the optimum adaptation of users is addressed. Following the RAT selection, the dynamic device adaptation to the new RAT should take place, to cope with application and QoS requirements. For example, after a change in the RAT, an update in a protocol component/codec may be triggered (either network initiated or device initiated) for various reasons: i) to compensate for QoS degradation, ii) to provide a protocol patch update to fix a software bug iii) to provide a new version of an existing component with enhanced capabilities. In this sense, a generic framework is provided that handles the necessary mechanisms for downloading, installation and on-the-fly activation of missing protocol-related RAT components. The following subsections highlight the focus and design assumptions in each of the previously mentioned reconfiguration phases.

2.1 RAT Selection Context

The RAT selection phase addresses an efficient attribution of corresponding resources to a specific user (different RATs such as WiMAX, WiFi (IEEE802.11a/b/g/n, etc.), 3GPP, DVB-T or DAB, different bands, etc.) in a distributed system, minimizing the required complexity in the network and user side as well as the signaling overhead. The focus is laid on techniques that are fully compatible with legacy technologies; the proposed approaches are also applicable to future air-interfaces, following the trend for the deployment of a (physical or virtual) cognitive channel as a single new element to be exploited for finding optimum resource usage strategy. These approaches are meant to be transparent to the physical user – any reconfiguration process is handled automatically by the equipment devices.

In addition, each terminal/user can apply several strategies in order to get the best service requested by the user. Multi-mode and reconfigurable terminals have the capability to connect simultaneously to several wireless network resources and also to reconfigure themselves in order to connect to new radio access technologies available in a cell. Given that multi-mode and reconfigurable network equipments inherently provide enhanced capabilities (by either dynamically adapting a specific radio access resource, or by reconfiguring some nodes to dynamically provide higher system capacity, depending on demands in a given area), consequently, the terminals should automatically adapt to the new scenario.

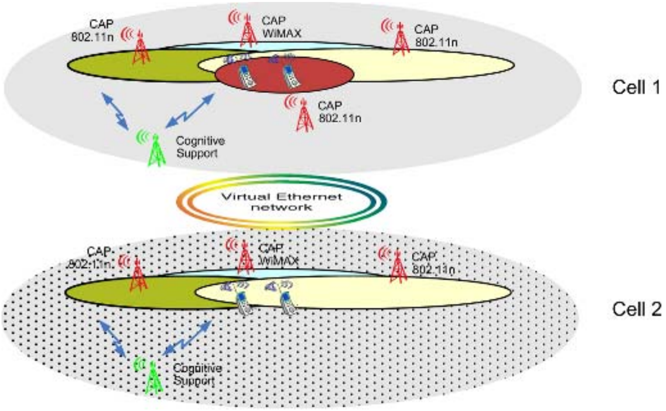


Fig. 1. A distributed network approach in a multicell context with different Cellular Access Points (CAPs)

Moreover, it is assumed that the system is organized in an entirely distributed way: the network propagates "policies" (e.g., via the Cognitive Channel) which define generic behavioral rules to be applied by any network and user equipment. Consequently, the network/user equipment is NOT parameterized by any central controller, but adapts autonomously (typically applying "Autonomic Networking" principles) to the constantly changing environment. This finally leads to a distributed optimization of the resource use. In the same example, a possible environmental change triggering user adaptation is illustrated: a RAT terminates its services and the remaining resources (RATs, bands, etc.) must thus be split among the active users.

The problem addressed within this paper concerns the optimum adaptation of users to a changing context/environment using autonomic networking and policy-based self-governance principles. A mechanism is proposed that enables users to adapt their resource use autonomously (applying autonomic networking approaches and relying on policy based self-management), such that a suitable compromise is found that is near-optimum from the perspective of a specific user ("*get maximum data rate, even if I penalize other users*") and from the network perspective ("*maximize total network throughput and split resources fairly among all users*").

2.2 Protocol Reconfiguration

Following the RAT selection, the protocol reconfiguration phase is aimed to address generic mechanisms for the deployment of transparent plug-in of protocol components in equipments. The presented solution is aligned with a set of assumptions regarding the design aspects of the proposed architecture and mechanisms:

- a protocol stack is composed of discrete protocol layers. The communication between them is established either using standard defined interfaces, i.e. Service Access Points (SAPs) or queue-based communication schemes. This design also facilitates the maintenance of cross layer optimization issues in the protocol stack. In addition, this design provides the capability of specifying a protocol stack according to application needs, QoS requirements as well as the specific RATs.
- A protocol layer is composed of protocol components. Each protocol component may specify specific protocol functionality (i.e, if we consider a TCP protocol, a TCP component may realize the congestion control algorithms) or a combination of different functionalities (i.e, a TCP component that realizes both congestion control and flow control algorithms).

The introduced framework based on the above considerations is aimed to cope with the following protocol reconfiguration aspects: the dynamic binding of component services into a fully fledged protocol service as well as the runtime replacement of protocol functionality. Specifically, this solution extends the typical Manager-centric architectures for the establishment of component bindings introducing a distributed model. Such model apportions the above mentioned functionality to the protocol components. The latter is based on a semantic-layer of information which describes static characteristics of the components as well as dynamic characteristics to capture the environment configuration.

The above analyzed mechanisms are incorporated into a generic management and control architecture enabling dynamic protocol reconfiguration via self-configuring protocol components (Fig. 2). In particular, the following key elements are identified:

- The Download Manager module which caters for the software download in the system, as well as for authorization procedures and integrity checks.
- The Installation Manager, which is responsible for post-download steps as well as the software installation to the system.
- The Decision Manager module which specifies concrete decision concerning reconfiguration actions, based upon a set of policy rules and contextual

information. In the scope of this paper, such module is responsible for the protocol stack configuration, in terms of specifying the different protocol layers and components to be used, as well as for triggering a protocol stack update.

- The Autonomic Manager module, which is responsible for the overall monitoring and control of the software operation, i.e., it instantiates the various components/triggers the component replacement process.

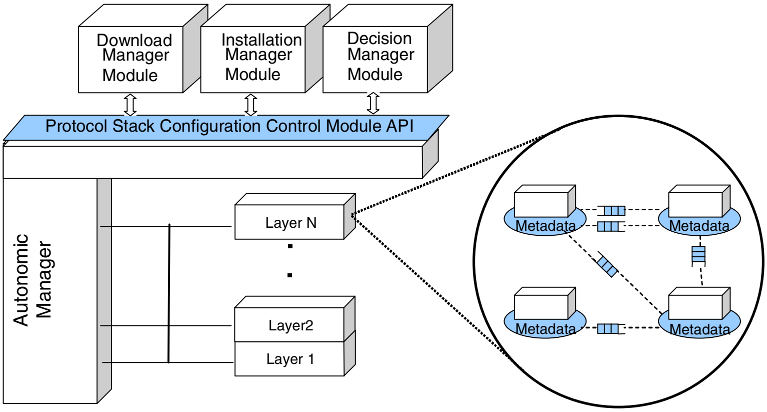


Fig. 2. A Management and Control Architecture Enabling Self-Configuring Protocols

3 Analysis of Game Theory Based RAT Selection in a Simple Two-User Context

Considering a simple two-user scenario, this section illustrates the application of a game-theoretic analysis [2] in order to derive suitable policy rules directing the user behavior. It is shown that global policies, applicable to all users, reduce the RAT selection convergence time considerably. Moreover, a *global* policy assures a minimum signaling overhead, since the user terminals are not addressed independently as it is the case of a centralized approach. The main aspects presented below can be extended to more complex scenarios consisting of a multitude of heterogeneous RATs and a multitude of users at the cost of an increased complexity for the RAT selection and search for suitable policies. This generalization, however, is out of the scope of this paper and will be discussed in future contributions.

3.1 Scenario Definition

The following scenario is considered in this analysis: An operator controls four IEEE802.11n Access Points (APs), each operating in a distinct 20MHz band and at a distinct carrier frequency. There are two Mobile Terminals (MTs) communicating over 1, 2, 3 or all of the available bands. The operator decides to switch off one AP, and indicates this information by propagating a corresponding message to the MTs. The MTs then need to redefine their spectrum / AP use autonomously. Each MT has the choice among seven possible spectrum allocation strategies denoted from S_1 to S_7:

1) S_1: use band #1;
2) S_2: use band #2;
3) S_3: use band #3;
4) S_4: use bands #1 and #2;
5) S_5: use bands #2 and #3;
6) S_6: use bands #1 and #3;
7) S_7: use bands #1, #2 and #3.

A simplified throughput computation model is used, assuming a throughput per band (or channel) equal to D bit/s. When a given channel is reserved to only one MT, the total throughput D is available for the MT. In case it is split among two MTs, the total throughput decreases due to collisions: D' = D*d where 0<d<1 is a kind of penalty factor, and each MT gets a throughput of D'/2 = D*d/2 with 0<d<1. In the examples below, we choose "d=0.9" for illustration purposes. The issue addressed is to find the best combination of strategies for both MTs such that maximal throughput is achieved for both.

3.2 Performance Analysis

The analysis is carried through the 2D game table presented below; the rows and the columns correspond to the strategies of MT1 and MT2 respectively. In addition, the table elements correspond to pairs of throughput values (MT1 throughput, MT2 throughput), obtained when MT1 and MT2 are using a given combination of strategies.

Table 1. Overall game table (1st column: User 1 strategies, 1st line: User 2 strategies)

	S1	S2	S3	S4	S5	S6	S7
S1	**(0.45, 0.45)**	**(1,1)**	**(1,1)**	**(0.45, 1.45)**	**(1,2)**	**(0.45, 1.45)**	**(0.45, 2.45)**
S2	**(1,1)**	**(0.45, 0.45)**	**(1,1)**	**(0.45, 1.45)**	**(0.45, 1.45)**	**(1,2)**	**(0.45, 2.45)**
S3	**(1,1)**	**(1,1)**	**(0.45, 0.45)**	**(1, 2)**	**(0.45, 1.45)**	**(0.45, 1.45)**	**(0.45, 2.45)**
S4	**(1.45, 0.45)**	**(1.45, 0.45)**	**(2, 1)**	**(0.9, 0.9)**	**(1.45, 1.45)**	**(1.45, 1.45)**	**(0.9, 1.9)**
S5	**(2, 1)**	**(1.45, 0.45)**	**(1.45, 0.45)**	**(1.45, 1.45)**	**(0.9, 0.9)**	**(1.45, 1.45)**	**(0.9, 1.9)**
S6	**(1.45, 0.45)**	**(2, 1)**	**(1.45, 0.45)**	**(1.45, 1.45)**	**(1.45, 1.45)**	**(0.9, 0.9)**	**(0.9, 1.9)**
S7	**(2.45, 0.45)**	**(2.45, 0.45)**	**(2.45, 0.45)**	**(1.9, 0.9)**	**(1.9, 0.9)**	**(1.9, 0.9)**	**(1.35, 1.35)**

To give an example: in the first cell on the upper left corner, user 1 chooses "strategy S1" and user 2 equally chooses "strategy S1"; in conclusion, both users are sharing a single channel where collisions may occur and the throughput per user is ½*d = 0.45. After analyzing the game table, the existence of a unique Nash equilibrium when both users choose strategy S_7 (red cell) is apparent. In fact, this forms a stable state which no user would find it interesting to deviate from. However, it is not Pareto efficient since better couples of throughputs are obtained with other combinations (yellow cells).

If a given user follows the simple rule of always targeting the maximal throughput, no matter what are the consequences on the other user, he will choose strategy S_7 and reach the states corresponding to the green cells; this situation results in an operating point which is suboptimal, in spite of being a Nash equilibrium.

For instance, suppose users play in turn, as represented with the orange arrows in Table 1. If users are in an initial state such that both users pick up strategy S_1 (the normalized throughput they both achieve equals to 0.45) and if user 2 is the first to *play, he will try to achieve the maximal throughput and therefore chooses strategy S_5 (he achieves throughput equal to 2 instead of 0.45). Then given the new strategy of user 2, user 1 will try to maximize its throughput in turn and chooses strategy S_7 (the normalized throughput he achieves equals to 1.9 instead of 1). Finally, user 2 responds by selecting strategy S_7 and the equilibrium is reached, since both users achieve throughput equal to 1.35 and no one can improve its throughput by modifying only its own strategy.*

3.3 Derivation of Suitable Policies

The idea is to establish controlled competition so as to get the fairest split of resources and reach the states corresponding to the yellow cells. This is achieved by the use of simple policies propagated by the operator, e.g. "do not use strategy S_7".

The operating point search is made on the following suitable where strategy S_7 has been removed for both users. If the game is played based on this table (Table 2) and users still follow the simple rule of always seeking for the maximal throughput (no matter what are the consequences on the other user), the states corresponding to the yellow cells will systematically be reached.

For example, suppose again that users play in turn, following the orange arrows represented on Table 2. If users are in the same initial state as previously (both users select strategy S_1 and achieve normalized throughput equal to 0.45) and if user 2 is the first to play, he will choose strategy S_5 (he achieves a maximal throughput equal to 2 instead of 0.45). Then given the new strategy of user 2, user 1 will try to maximize its throughput in turn and chooses indifferently strategy S_4 or S_6 to get 1.45 instead of 1. Since the resulting throughput of user 2 is also maximized (he cannot achieve better throughput than 1.45), this new configuration is an equilibrium, which is clearly more efficient than the previous equilibrium where users both picked up strategy S_7.

Table 2. Modified game table (1st column: User 1 strategies, 1st line: User 2 strategies)

	S1	S2	S3	S4	S5	S6
S1	**(0.45, 0.45)**	**(1,1)**	**(1,1)**	**(0.45, 1.45)**	**(1,2)**	**(0.45, 1.45)**
S2	**(1,1)**	**(0.45, 0.45)**	**(1,1)**	**(0.45, 1.45)**	**(0.45, 1.45)**	**(1,2)**
S3	**(1,1)**	**(1,1)**	**(0.45, 0.45)**	**(1, 2)**	**(0.45, 1.45)**	**(0.45, 1.45)**
S4	**(1.45, 0.45)**	**(1.45, 0.45)**	**(2, 1)**	**(0.9, 0.9)**	**(1.45, 1.45)**	**(1.45, 1.45)**
S5	**(2, 1)**	**(1.45, 0.45)**	**(1.45, 0.45)**	**(1.45, 1.45)**	**(0.9, 0.9)**	**(1.45, 1.45)**
S6	**(1.45, 0.45)**	**(2, 1)**	**(1.45, 0.45)**	**(1.45, 1.45)**	**(1.45, 1.45)**	**(0.9, 0.9)**

At this point it should be pointed out that the use of a very fundamental policy rule expressing a constraint on the strategy selection ("do not use strategy S_7"), makes it possible to avoid sub-optimal Nash equilibrium.

3.4 Analysis of Impact of Policy Introduction

In the following figures, the game evolution in four different scenarios is illustrated:

- Scenario 1: users play simultaneously and do not consider policy rules;
- Scenario 2: users play simultaneously and respect the previously mentioned policy rule;

- Scenario 3: users play in turn (i.e. one after each other) and do not consider policy rule;
- Scenario 4: users play in turn and respect the previously mentioned policy rule.

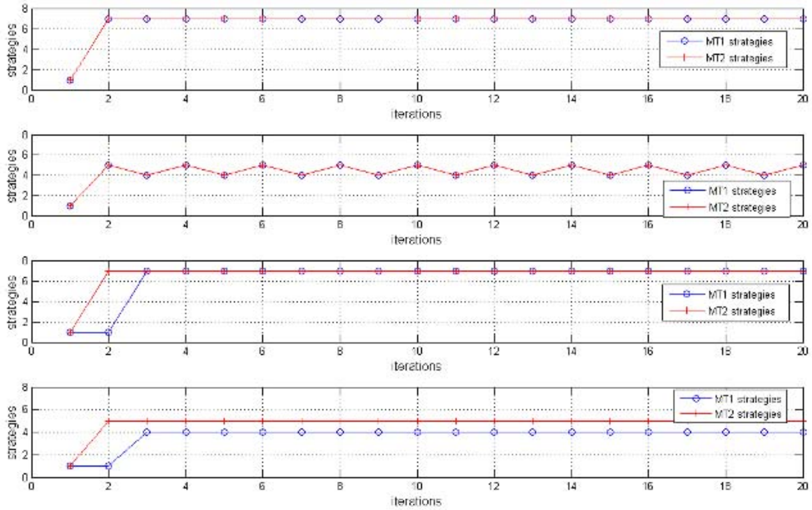


Fig. 3. Convergence of strategies

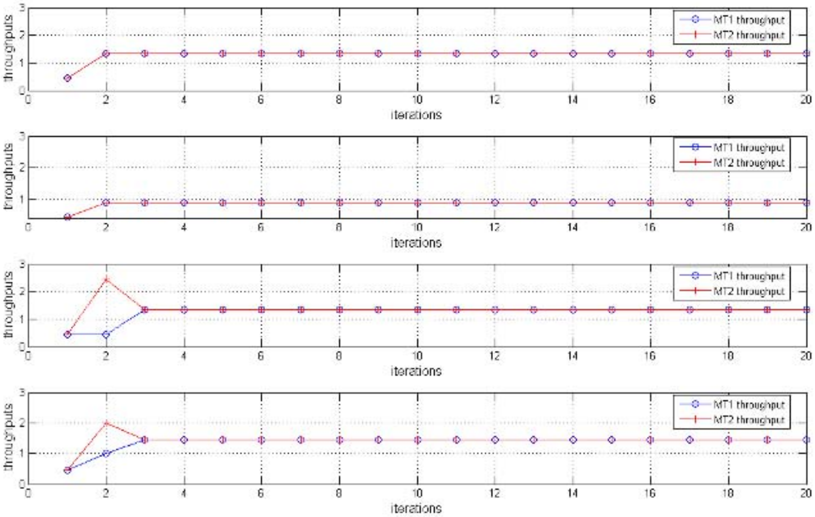


Fig. 4. Data Rates

The first series of four graphs (Fig. 3) represents the evolution of the users' choice of strategy for the set of four scenarios, while the second (Fig. 4) represents the users' throughput evolution for the same four scenarios. It should be noted that a configuration where users play in turn converge to the equilibrium faster than when users play simultaneously. Moreover, the curves confirm that the use of an appropriate policy rule helps the system to converge towards an absolutely efficient equilibrium.

4 A Framework for Dynamic Protocol Reconfiguration over Heterogeneous RATs

This section highlights the procedure of protocol reconfiguration, describing the proposed framework and mechanisms through a reconfiguration scenario over heterogeneous RATs.

4.1 Protocol Reconfiguration

The section analyzes the fundamental phases of the protocol reconfiguration procedure, Firstly, the mechanisms involved in the protocol stack bootstrap are considered. In addition, a simple case that the decision mechanisms embedded in the terminal dictate that a protocol reconfiguration should take place is presented. This decision concerns the downloading, installation and on-the-fly activation of a protocol component.

Control Signaling for Protocol Stack Bootstrap
During the protocol stack bootstrap, a configuration of the protocol stack is selected by the autonomic decision making functionality; such configuration specifies the protocol layers that should form the protocol stack as well as the components that should be used within each protocol layer. Thereafter, the Decision Manager informs the Autonomic Manager about the protocol layer configuration and the protocol component configuration. After acknowledging the reception of this information, the Autonomic Manager proceeds with the instantiation of the protocol components selected for each protocol. Considering that the binding between the protocol layers is realized via the standard defined Service Access Points (SAPs), the focus is on the procedures related to the protocol component binding and replacement. Therefore, the reconfiguration signalling depicted in Figure 1 illustrates only the instantiation of two protocol components that form a protocol layer (Component TestA and Component TestB) [16].

Semantic-Based Dynamic Binding of Protocol Components
Next, the semantic-based dynamic binding of protocol components is performed. Specifically, the components evaluate the dynamic characteristics of their metadata and identify the components there are composed with. Finally they establish the bindings to the components they are composed with.

The details of this procedure are illustrated with an example; a protocol layer comprised of two autonomic protocol components (CompA and CompB) is considered and the dynamic binding of these components is analyzed. In this case study a unidirectional communication pattern between the protocol components is assumed, in the sense that CompA sends data to CompB. The XML representation for the metadata profiles of CompA and CompB is illustrated in Fig. 5 (a) and (b) respectively.

The metadata profiles include static characteristics specifying the component identification (name, version, path to source code), whereas the dynamic characteristics depict their current configuration in the system and are dynamically updated according to the different protocol stacks stratification (dynamic characteristics include arrays for input and output components and their static characteristics). For example, as depicted in Fig. 5 (a), the protocol component CompA does not provide any input interface, whereas it provides an output interface to CompB.

Based on the interpretation of the metadata profile, the component composition is realized. During this phase at first, CompA checks the input array in its metadata, which does not include any components. Thereafter, it checks its output array, which includes CompB. Next, it should verify the composition between CompA and CompB by checking that the input array in CompB metadata includes CompA. The same procedure is applied by CompB. At this point it should be clarified that this procedure also applies for all the protocol components regardless of the number of components they are composed with.

```
 <?xml version="1.0"
encoding="ISO-8859-1" ?>

<component>
    <id> CompA</id>
    <version>Version 1</version>
    <path>/CompA</path>

    <inputs>
            <input No="0" />
    </inputs>

    <outputs>
         <output No="1">
         <id>CompB</id>
         <version>Version 1</version>
         <path>/CompB</path>
         </output>
    </outputs>

 </component>
```

(a)

```
 <?xml version="1.0" encoding="ISO-
8859-1" ?>

<component>
    <id> CompB</id>
    <version>Version 1</version>
    <path>/CompB</path>

    <inputs>
         <input No="1" />
         <id>CompA</id>
         <version>Version 1</version>
          <path>/CompA</path>
    </inputs>

    <outputs>
         <output No="0" />
    </outputs>

 </component>
```

(b)

Fig. 5. Metadata profiles for protocol components CompA and CompB

After the validation and verification of the composition, the component communication establishment is realized. Specifically, each component establishes a communication link for each component it is composed with by creating a FIFO queue. The latter is realized with the use of a unique key. Such key is generated by a conversion function that produces a global unique output based only on a unique parameter, the concatenated String of the ID of the specified component and the ID of the component it is composed with. In addition, it should be noted that with the use of this ID, the specified component creates or accesses a FIFO queue, depending if it already exists in the system. The composition verification and establishment procedures are repeated for all the protocol components that are bound to the specified component.

Control Signaling for Protocol Reconfiguration

This phase comprises the necessary steps for the protocol reconfiguration process. Such procedure may be triggered by a change in the environment (new RAT/cellular system, handover procedures), or the QoS requirements posed by applications or user preferences. Next, the decision module specifies the new protocol stack configuration. Based on contextual information about the protocol stack, it identifies whether the specified components exist in the system. In case of missing components, as in the scenario in Fig. 6 the Decision Manager should select the appropriate protocol component from the repository available in the network. Thereafter, the Autonomic Manager requests the Download Manager to perform the component downloading. After the successful realization of this procedure, the component installation in the system is performed by the Installation Manager. Finally, the Decision Manager informs the Autonomic Manager about the new system configuration.

Regarding the selection of the most appropriate protocol component from the network, a theoretical approach is introduced. At first the interface compatibility between the stationary and available components is checked; then the compatible components are compared in order to select the "best", based on the QoS it may provide and the delay introduced for its downloading in the system.

This analysis is based on a modular system with the following formulation [8]. Let us assume there are i protocol components, with input-output interfaces. In this sense, compatible components have the same interfaces with the same components. The methodology to find the compatible components is based on the Graph Theory.

Every group of compatible components belongs to a module s_i. Further, $I = \{s_1, s_2, s_3\}$ is the set of these modules, s_i module comprises all the compatible components. Each module is associated with: 1) a vector of output variables $P_{s_i} \in P_i$, which describes the output interfaces of the protocol components. Each module has an initial estimate P_i^0 of its vector, which defines the output interface of the initial protocol components connection. 2) a set of input interfaces vectors from it's neighbour modules $P'_{s_i} \in P'_i$ and N_i which specifies the modules with which modules belonging to I need to interact. 3) An objective function Q_i: that is a measure of how well the output vector P_i of the specified module satisfies the task of the module, given its inputs from all its surrounding modules:

$$Q_{s_i}(P_{s1}, P_{s2}, P_{s3}) = Q(P_{s_i}), \quad Q_{s_i}(P_{s1}, P_{s2}, P_{s3}) = q_i(P_i) + \frac{1}{\lambda_i g_i(P_i)},$$

where q_i is the quality of service function of each protocol component, g_i is the delay function of each protocol component and λ_i depends on the number of the replaced components.

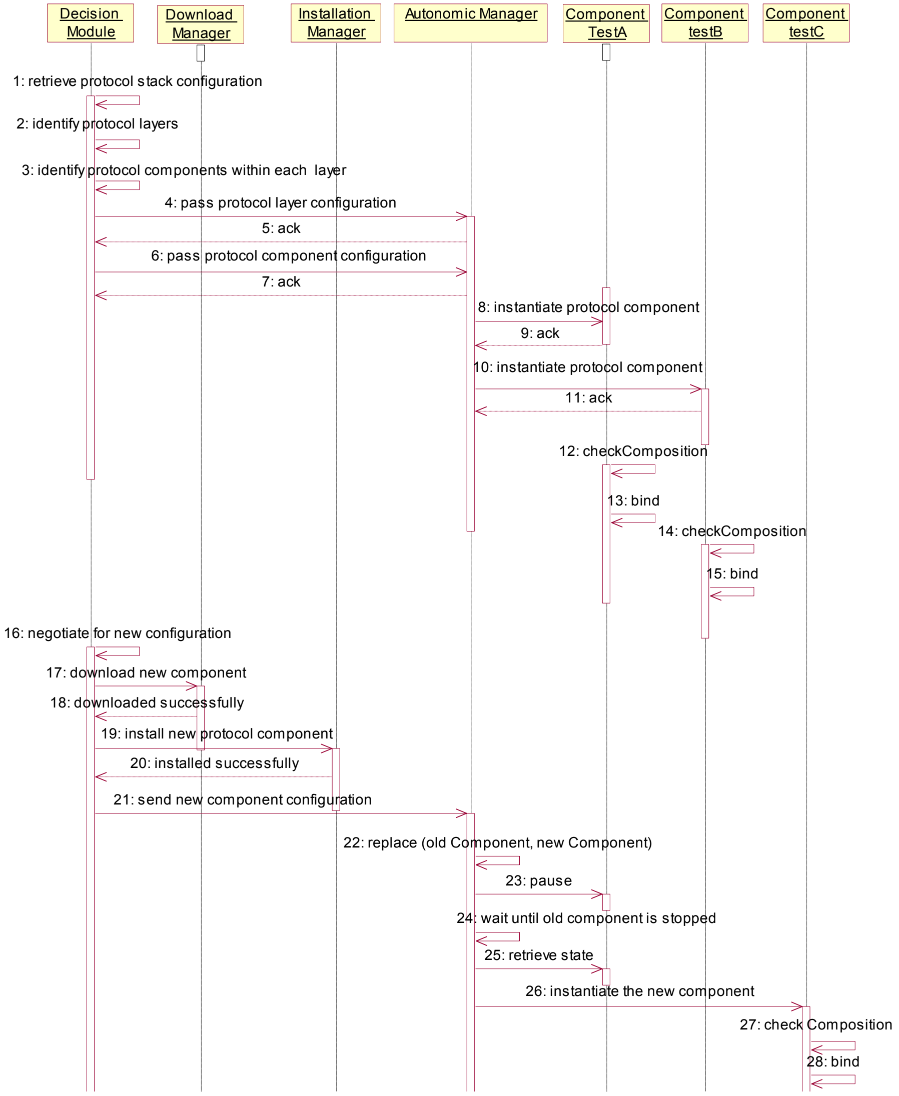

Decision Module
Download Manager
Installation Manager
Autonomic Manager
Component TestA
Component testB
Component testC
1: retrieve protocol stack configuration
2: identify protocol layers
3: identify protocol components within each layer
4: pass protocol layer configuration
5: ack
6: pass protocol component configuration
7: ack
8: instantiate protocol component
9: ack
10: instantiate protocol component
11: ack
12: checkComposition
13: bind
14: checkComposition
15: bind
16: negotiate for new configuration
17: download new component
18: downloaded successfully
19: install new protocol component
20: installed successfully
21: send new component configuration
22: replace (old Component, new Component)
23: pause
24: wait until old component is stopped
25: retrieve state
26: instantiate the new component
27: check Composition
28: bind

Fig. 6. Signaling for Protocol Stack Bootstrap and Protocol Self-Configuration

The decision procedure, in terms of replacing a compatible component with the one can be modelled as a game. Specifically, a game with $I = \{s_1, s_2, s_3\}$ players is considered; in our example the game has three players s_1 , s_2 and s$_3$. Each player is associated with a decision vector (An individual decision vector P_{s_1}, P_{s_2} and P_{s_3}) and a payoff function. During the course of the game, which is a sequence of stages and moves, each player chooses a specific decision vector. The payoff function Q evaluates the performance of the player based on its decision P_{s_i} and the decisions from the other players that influence the decision of player i. The I player game starts; each player wants to find the decision that optimises its payoff. In particular, a solution requires: 1) a concept of what is meant to be optimality 2) decision making models that allow for the computation of this equilibrium. There exist no cooperation among the players and each player makes its own decision independently. The game theoretic integration framework is a particular solution to the integration problem where the decision making model is defined in the context of no cooperative games.

Dynamic Replacement of Protocol Components

After evaluating the new protocol configuration, the Autonomic Manager should perform the replacement of the old protocol component with the new one. At first, the Autonomic Manager pauses the functionality of the component under replacement and retrieves its execution state (Fig. 6). Next, the Autonomic Manager instantiates the new component and dispatches to it the retrieved state information. Based on the acquired state information and its metadata, the new component realizes its dynamic composition with existing software components (by accessing the FIFO queues that correspond to existing communication links). The above analyzed on the fly replacement process also allows the reliable operation of the software under configuration, since it applies state management models to ensure the transparent switching from the old to the new component.

4.2 Validation and Performance Assessment

Targeted to verify and validate the introduced framework, a proof-of-concept prototype is implemented, concerning the dynamic binding and reconfiguration capabilities of test protocol components. Specifically, this prototype concerns the deployment of the following functionality: a) the Autonomic Manager, b) a test protocol which comprises of two protocol components (CompA and CompB presented in the example) with unidirectional communication as well as the component metadata and c) the presented reconfiguration framework upon its application in the test protocol.

Exploiting the mechanisms and procedures presented in the previous sections, the implemented Autonomic Manager initiates these two protocol components; thereafter the protocol components discover and access their metadata files and automatically establish their bindings. In addition, a simple reconfiguration scenario was realized, in terms of replacing CompB with another component with similar functionality (CompC), during runtime operation of this test protocol. Following the scenario presented in Fig. 6, the dynamic replacement of CompB was achieved; in addition the new component incorporated itself seamlessly by taking into account the current system composition and configuration. Furthermore the reliable operation of the test protocol was preserved during the replacement process, in terms of achieving no loss of protocol data or existing connections. Therefore the process of autonomic component self-configuration (dynamic component binding and replacement) was successfully validated under the current prototype, thus proving that the protocol components, when reassembled, form the initially specified protocol functionality.

Furthermore, the performance assessment of the proposed system was considered taking into account that in communication devices, the protocol stack subsystem must meet strict performance requirements. Another aspect that should be taken into consideration is the overhead introduced by the incorporation of autonomic features in the new framework, in terms of performance metrics. In order to address these issues and validate the feasibility of the autonomic approach, performance evaluation studies were executed considering the following factors:

- the calculation of the overhead that is created for the dynamic establishment of bindings between the protocol components, due to the introduced of autonomic capabilities,
- the calculation of the time that is necessary for the specified protocol reconfiguration to take place

The delay associated to the reconfiguration of the protocol was defined as the time interval from the time instance the existing protocol component is being signalled by the Autonomic Manager to stop its operation until the time instance the new protocol component is successfully incorporated in the system. The time interval for the original framework includes the following operations:

- Autonomic Manager waiting in idle state until the old component stops (so as the old component is in a safe state at the time of replacement),
- Instantiation/initialization of the new protocol component by the Autonomic Manager,
- Self-configuration of the new protocol component (the new protocol component identifies and establishes its bindings with the existing protocol components on its own)

In particular, Fig. 7 (a) illustrates the binding delay for each component within a set of 100 samples, considering the underlying hardware capabilities (Pentium III 800MHz PC with 512MB RAM) and operating system (Debian Linux 3.0 R4). The mean value for this binding delay is 787,46 μsec for CompA and 462,89 μsec for CompB. The minimum value of the binding delay is found to be 505 μsec for CompA and 415 μsec for CompB, whereas the maximum value is 2039 μsec and 694 μsec for CompA and CompB respectively (it should be noted that the delay is greater for CompA compared to CompB, since the Autonomic Manager firstly initiates CompA; this way CompA creates the communication queue, whereas CompB simply accesses it).

Moreover, Fig. 7 (b) illustrates the delay that was introduced for the protocol reconfiguration, for a set of 100 samples, considering the aforementioned experiment setup capabilities. The mean value for this delay is 2381 μsec (the minimum and maximum values are 2284 and 2456 μsec correspondingly).

The performance evaluation studies proved that the deployment of the proposed framework and the introduction of autonomic capabilities in the protocol components have minimum performance impact in the system, thus increasing its flexibility.

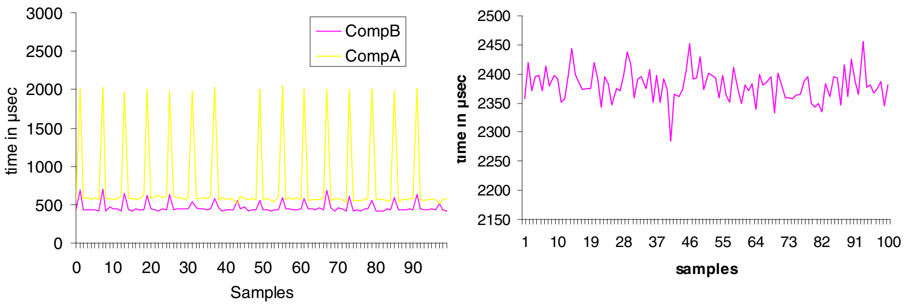


Fig. 7. (a) Binding Delay for the autonomic components within a set of 100 samples, (b) Total time for the replacement of the old component with the new one

5 Related Work

The vision for autonomic computing and communications was the basis for several research activities in the past years in both industry and academia. These activities spawn across the definition, design and deployment of self-* features in emerging communication systems and devices [6]. Following the model proposed by IBM [5], an autonomic system should at least incorporate four attributes: self-configuring, self-healing, self-optimizing and self-protecting, known as self-CHOP features. On the other side, additional features for autonomic communications systems are addressed in [7], including self-awareness, self-adaptation, self-implementation and self-description.

In addition, several work was performed in the context of deploying the above presented self-* features. Since one of the basic capabilities addressed in autonomic communication environments is the autonomic decision making, previous work in policy based management was considered. In particular, the approach presented in [9] should be mentioned, which addresses the policy management issue from a new perspective through posing it as a problem of learning from current system behavior, while creating new policies at runtime in response to changing requirements. A hierarchical policy model is used to capture users and administrators' higher level goals into network level objectives.

Moreover, regarding the introduction dynamic configuration or autonomic capabilities in software subsystems, several other approaches extend the conventional software design. The above concept was initially applied to X-kernel, Cactus and Appia frameworks, which deal with protocol composition based on microprotocol objects so as to fulfill the application QoS requirements [13][14] . In addition, the CORBA Component Model specifies components for distributed software systems, However it is not appropriate for use in autonomic environments due to its limitations regarding standard defined interfaces and limited extension of object functionality [10][11]. Moreover, the Accord Programming Framework enables the development of autonomic components, but gives limited information on the design blueprints, pertaining to the establishment of component composition and replacement [12].

Furthermore, a different vision on protocol stack design for autonomic comm.-unication based on the POEM model is analysed in [15]. This cross-layer design focuses on the advantages of layering and the benefits of holistic and systematic cross-layer optimization is at the core of this work. Finally, a Java-based protocol suite that supports protocol subsystems is analyzed in [17]. This system introduces great performance overhead (10:1) and does not specify the mechanisms required to achieve on-the-fly protocol reconfiguration.

Regarding the game theoretical analysis, it should be noted that it has recently attracted much attention in the general context of resource allocation in wireless networks. Moreover, it can be seen as an appropriate optimization tool for a fully distributed and scalable implementation (with a complexity transferred and shared out on the terminal side), where traditional centralized decision becomes computationally infeasible as the number of terminals in a cell, or the number of carriers in a multicarrier setting grows. Game theory has been applied to many wireless network problems, related to the physical, medium access or higher layers. In those problems, users are generally competing for a limited resource while having a limited knowledge of their environment, and therefore the strategic non-cooperative game model is often used, where each player selfishly tries to maximize its own utility regardless of the consequences its choice may have. For example, [3] deals with a power control game for wideband (e.g. CDMA-based) systems. Random access protocols, in particular ALOHA, are addressed in [4].

6 Conclusion

This paper presented a framework to cope with RAT selection and the requirement for transparent plug-in of protocol-related RAT components in heterogeneous systems. The study results illustrate that a suitable introduction of policies applicable by user terminals impact the system behaviour considerably. The simple example in a two-user context proved, that the interdiction of one strategy (applicable to all users) avoids sub-optimum equilibria and leads to a quasi-immediate convergence in terms of RAT selection strategy. As a follow-up of this work, the corresponding behaviour needs to be analyzed in more complex scenarios consisting of an increased number of users and RAT selection choices.

In addition, a protocol reconfiguration framework was analyzed, which provides the necessary mechanisms for component-based protocol reconfiguration. The signaling for the four basic phases of this procedure was specified as well as the mechanisms that enable the transparent protocol component self-configuration. Finally the validation of the proposed framework proved its feasibility, thus the introduction of minimum performance overhead in the system.

Acknowledgments. This work has been performed in the framework of the IST project IST-2003-507995 E2R II [18], which is partly funded by the European Union. The authors would like to acknowledge the contributions of their colleagues.

References

1. J. Strassner, "Autonomic networking – theory and practice", In Proc. *9th IFIP/IEEE International Symposium on Network Management* (*IM'2005*), Nice, France, May 2005.
2. D. Fudenberg and J. Tirole, "Game Theory", MIT Press, ISBN 0-262-06141-4, USA, 1991.
3. C.U. Saraydar, N.B. Mandayam and D.J. Goodman, "Efficient Power Control via Pricing in Wireless Data Networks," *IEEE Trans. on Communications*, Feb. 2002.

4. A. B. MacKenzie and S.B. Wicker, "Selfish users in Aloha: a game theoretic approach," In Proc. *Vehicular Technology Conference (VTC)*, vol. 3, Oct. 2001.
5. Jeffrey O. Kephart, David M. Chess. "The Vision of Autonomic Computing," Computer, vol. 36, n° 1, pp. 41-50, Jan. 2003
6. Murch, R., Autonomic Computing, Prentice Hall, 2004.
7. M.Smirnov, "Research Agenda for a New Communication Paradigm", Fraunhofer FOKUS White Paper, Nov. 2004, http://www.autonomic-communication.org/publications/doc/WP_v02.pdf
8. H. Isil Bozma and James S. Duncan, "A game Theoretic Approach to Integration of Modules", IEEE Trans. Pattern Anal. Machine Intell., vol. 16, no. 11, November 1994
9. N. Samaan and A. Karmouch, "An Automated Policy-Based Management Framework for Wired/Wireless Differentiated Communication Systems" in special issue of JSAC on Autonomic Communication systems, December 2005, Volume 23, Number 12
10. Wang, N., Schmidt, D.C., O'Ryan, C. Overview of the CORBA Component Model. In Component-Based Software Engineering: Putting the Pieces Together, Addison Wesley, Editors G.T. Heineman and W.T. Council. 2001.
11. OMG CORBA Components, version 3.0, June (2002), http://www.omg.org/
12. H. Liu and M. Parashar. A component based programming framework for autonomic applications. In Proceedings of the International Conference on Autonomic Computing, New York, NY, 2004.
13. Norman C. Hutchinson and Larry L.Peterson: The x-kernel: An architecture for implementing network protocols. IEEE Transactions on Software Engineering, 17(1):64-76, January 1991.
14. Sergio Mena, Xavier Cuvellier, Christophe Gregoire, Andre Schiper: Appia vs. Cactus: Comparing Protocol Composition Frameworks. 22nd International Symposium on Reliable Distributed Systems (SRDS'03), Florence, Italy, October 2003.
15. X. Gu, X. Fu, H. Tschofeni, L. Wolf, "Towards Self-Optimizing Protocol Stack for Autonomic Communications: Initial Experience", in the Proceedings of the 2nd IFIP International Workshop on Autonomic Communication (WAC'05), Athens, Greece, October 2005.
16. E.Patouni, N.Alonistioti "A Framework for the Deployment of Self-Managing and Self-Configuring Components in Autonomic Environments", in the Proceedings of the International IEEE WoWMoM Workshop on Autonomic Communications and Computing (ACC 06) Niagara-Falls, Buffalo-NY, 26-29 June 2006.
17. B.Krupczak, K.L.Calvert, M.A.Ammar, Implementing Communication Protocols in Java, IEEE Communications Magazine, October 1998.
18. End-to-End Reconfigurability (E2R II), IST-2003-507995 E2R II, http://www.e2r2.motlabs.com

Managing Policies for Dynamic Spectrum Access

David Lewis, Kevin Feeney, Kevin Foley, Linda Doyle, Tim Forde,
Patroklos Argyroudis, John Keeney, and Declan O'Sullivan

Knowledge & Data Engineering Group &
Centre for Telecommunication Value-chain Research,
School of Computer Science and Statistics,
Trinity College Dublin,
Dublin, Ireland
Dave.Lewis@cs.tcd.ie, Kevin.Feeney@cs.tcd.ie,
Kevin.Foley@cs.tcd.ie, linda.doyle@tcd.ie, timforde@mee.tcd.ie,
argp@cs.tcd.ie, John.Keeney@cs.tcd.ie,
Declan.Osullivan@cs.tcd.ie

Abstract. The advent of software radio technology and the resulting potential for dynamic access to the radio spectrum presents major new challenges in managing that access. These challenges arise from the likely spread of spectrum access decision-making authority well beyond existing regulatory authorities to a wide variety of co-existing market-based or open-access schemes. Policy-based management mechanisms are proposed as a flexible means for defining the rules that determine spectrum allocation dynamically. However, many existing policy based mechanisms rely on a fixed organisation structure and so are insufficiently flexible to support combinations of central allocation, market mechanisms and commons usage. In this paper we present the application of a novel policy-based management mechanism based on self-managing communities to the management of policy authoring authority. We show how an existing implementation could be used to manage a software-based radio system and how this approach provides self-organisation of multiple groupings with differing goals and policies in the allocation of spectrum. This is illustrated by taking real world policy authoring scenario from the world first software radio test license.

1 Introduction

The advent of software radio offers potential improvements in efficient use of the radio spectrum. By allowing agile, runtime reconfiguration of RF systems, spectral resources can be accessed dynamically, allowing applications to opportunistically exploit portions of the spectrum left unused at any point in time.

Currently the radio spectrum is largely allocated in a command and control manner based on long-range predictions of various licensees, e.g. TV broadcasters, military and emergency service users. This is coupled with the allocation of portions of spectrum to commercial licensees. In recent year some allocation has been on a competitive basis, e.g. through auction of 3G licenses, however the license terms are still long lived. This has led to under-utilisation of spectrum. It has also, arguably, slowed innovation in technologies for spectrum utilisation, as indicated by the

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 285 – 297, 2006.

acceleration of innovation in unlicensed bands, where utilisation technology is not linked a priori to the opportunity to access the spectrum.

To improve the utilisation of the spectrum, more dynamic access schemes are now being considered by regulators, in particular the F.C.C in the US and the U.K.'s Office of Communication (OfCom). This consideration is supported by advances in the development of agile radio technologies. In particular, there has been a recent trend toward performing more of the functionality in radio transmitters and receivers for data communication in software. Such software-based radio allows radio devices to dynamically switch between different bands, encoding schemes and protocols through software updates or runtime reconfiguration. Such reconfiguration could be made highly dynamic, as suggested for so called cognitive radio, where transmitters sense the spectrum for opportunities to make use of currently unutilised bands [mitola]. Cognitive radio executes a classic autonomic control loop in that it monitors its RF context, analyses the current opportunities then plans and executes a course of action, e.g. to access a particular band with specific power constraints, protocols and encodings. Consistent with this view of cognitive radio as an autonomic communications system, proposals are emerging for the governance of cognitive radio using policy-based management. The U.S DARPA Next Generation Communication (XG) project has proposed a policy-based management framework for cognitive radio [xg-vision]. This allows regulatory rules to be encoded as declarative rules that are executed at runtime as part of the autonomic control loop that implements cognitive radio.

However, central to the idea of dynamic spectrum access is not just that it leads to greater innovation in radio technology, such as cognitive radio, but that such innovation is linked to innovation of the use of the spectrum to meet commercial and social goals. As such, it should not be the aim of regulators to set the rules that are directly enforced in cognitive radio implementation. Instead, regulators will provide a framework of rules within which other bodies will be delegated authority to design the specific rules to which cognitive radio, or other software-defined radio schemes, will adhere. To maximise the opportunity to innovate in the combination of software radio technology and dynamic spectrum access policies a highly disaggregated approach to the delegation of policy-making authority should be encouraged. For example, policy-making authority could be delegated to municipal authorities to best meet local social conditions or different spectrum trading commodity markets could be established in different bands or regions to allow parallel experimentation with market rules. Such an approach may also result in value chains of policy-making authority, with secondary markets being established to satisfy niche trading requirements or regional sub-authorities being established, e.g. an airport being given policy-making authority by the municipal authority in which it is located. This will also allow more responsive tailoring of policies to local conditions. Tailoring can be tuned to geography, the bands subject to the authority, the commercial and social goals guiding policy-making, the density of users/terminals and the range and balance of applications using the spectrum in that locale.

In this paper we identify some shortcomings in the DARPA XG policy framework in its ability to handle a disaggregated ecology of policy making authorities. We propose how an existing scheme for community-based policy management can address these shortfalls and illustrate this with some case studies based on the real world policies needed under the world first software radio license that was recently allocated to the Centre for Telecommunication Value-chain Research (CTVR) by the Irish regulator, COMREG.

2 Policy Management of Software Radio

The DARPA XG Policy Language Framework aims to provide a means by which machine understandable policies can be defined in a highly flexible and traceable manner [xg-policy]. The design of the language aims to support the frequent changing of policies that apply to a particular radio system. These changes may be used to changes in operator policy, changes in the usage context of spectrum or mobility of the radio system between administrative domains where policies differ. In addition, the language aims to support consistency checking of policies and the easy introduction of new language concepts.

To achieve these goals the XG policy language incorporates many features from modern policy languages. One such innovation is the use of ontology-based semantics to capture machine-processable facts about element of policies. Following their successful application in policy languages such as Rei [kagal] and Kaos [uszok], the XG policy language uses a description logic approach for describing facts as standardised by the World Wide Web Consortium in the Web Ontology Language (OWL) [owl] as part of its Semantic Web initiative. Such facts define concepts for use in policies that can be inherited from multiple existing concepts, simplifying the modelling of spectral resources and the extension of such models over time. Such facts can also be used to define constraints over the concepts used in policies. Such OWL based facts allow existing description logic reasoners to be employed in checking the consistency of policies and inferring new constraints on the application of policies.

XG policies consists of a selector descriptor, which allows for easy filtering of relevant policies, an opportunity descriptor, which defines whether a valid opportunity exists and a usage constraint descriptor which defines constraints on the device and environment under which the opportunity can be exploited, e.g. maximum transmit power in the available band. The selector descriptor contains descriptions of the authority who is establishing and enforcing the policy, the frequency range or frequency group, the region and time over which the policy applies and the capabilities of an device able to enforce the policy.

By default all policy rules satisfying the selector for a device that experiences an opportunity and can conform to the opportunity constraints are applied. However, in common with other policy languages, the XG language allows the definition of meta-policies that specify rules about how other policy rules are enforced. In particular, XG allows policies to be groups, either explicitly or by satisfying a logical expression and it allows precedence of policies to be defined explicitly.

The illustrate the use of this language we specify the policy rules that capture the terms of a recent spectrum usage license granted by to the Centre for Telecommunications Value-chain Research (CTVR), a large multi-institute research effort in the Republic of Ireland (www.ctvr.ie), by the Irish communications regulator, COMREG. This license was granted explicitly for the testing of software radio and is believed to be the first such license to be granted worldwide. It is thus a fitting subject for analysing the suitability of the XG policy language. Below we use the CLIPS based notation from [xg-policy] to specify the policy rules that define the terms of this license.

No.	Encoded Policy in Shorthand Notation	Remarks

1	`(PolicyRule (id P1)`
	`(selDesc S1)`
	`(deny FALSE)`
	`(oppDesc AnyOpp)`
	`(useDesc U1) )`
		-- Overall rule defining COMREG license. The terms of the license are not concerned with opportunities.

2	`(SelDesc (id S1)`
	`(authDesc COMREG)`
	`(freqDesc F1)`
	`(regnDesc R1)`
	`(timeDesc T1)`
	`(devDesc D1) )`
		-- The descriptor defining the authority specifying to policy rule (COMREG). The breakdown of the time and device descriptor omitted for brevity

3	`(Usedesc (id U1)`
	`(xgx "(<=`
	`MaxTransmitPower`
	`TxParam)" ) )`
		-- Usage descriptor defines maximum transmit power

4	`(Power (id TxParam)`
	`(magnitude 1.0) (unit W)`
	`)`
		-- Maximum transmit power is 1W

5	`(FrequencyDesc (id F1)`
	`(frequencyRange bandA)`
	`(frequencyRange bandB) )`
		-- The license specifies two bands (details of band B omitted)

6	`(FrequencyRange (id band)`
	`(max 2.0925)`
	`(min 2.0675)`
	`(unit GHz) )`
		-- A 25MHz band centred on 2.08 GHz

7	`(RegionDesc (id R1)`
	`(region`
	`TrinityCollegeDublin)`
	`(region`
	`UniversityOfLimerick)`
	`(region`
	`DublinCityUniversity)`
	`(region`
	`UniversityCollegeCork) …)`
		-- Some of the names regions are omitted for brevity. The license simply implies that transmitted must be placed on the premises of these institutes so this does not necessarily match exactly the XG models for a geographical point coordinate or a cylindrical volume.

The above policy represents the extent of the license, and thus of the spectrum allocation rules as specified by COMREG. The license then delegates to the CTVR Executive administrative authority for how this is used. Though the DARPA XG policy language version 1 has some mention of the delegation of authority, allowing to be allocated using the fact keyword PolicyAdministrator, it defers guidance on how the delegation of such authority should be modelled to future version. It is the complexity of such delegation in the scenarios described in then previous section that we address in this paper.

Even in the relatively simple case of the CTVR software radio license, an analysis of current working structures in CTVR indicated some of the complications that may arise. For instance it would seem reasonable that the CTVR Executive, on receipt of the license give authority for administering the access to the licensed bands in each location to local network administrators. These local administrators may then define rules which indicate which users can access which sub-bands at which times. However, as a multi-institute centre there often emerges the need to support cross-institute research that may result in spectrum access policy conflicts between the proponent of that research and the local administrators. In a relatively cohesive organisation such as CTVR such conflicts can be resolved by direct communication, however unresolved conflicts can be escalated to the CTVR Executive as the organisational arbiter of access control policies. The Executive may itself possess rules for resolving such conflicts, e.g. simple majority of executive committee members. The capture and execution of this policy resolution process is not easily supported in the DARPA XG policy language currently, largely due to limited mean for expressing policies about the authority to define policies and the organisational modelling this requires. In the next section we examine an generic mechanism based on the idea of hierarchical communities of policy makers that could address this deficiency in DARPA XG. Such flexibility in resolving conflicts in spectrum access policies from different bodies is essential if we wish to allow innovation in mediating access, for instance by using market-based rules to resolve conflicts.

3 Community Based Policy Management

Community based policy Management is a new alternative to modelling a dynamic, adaptive system that moves away from the more traditional form of system modelling; role based modelling [feeney]. The system model, in the case of this paper, is concerned with policy determination for dynamic spectrum access. As indicated above, the DARPA XG policy language provides some basic support for grouping policies, for setting rules about policy execution precedence and offering alternatives between policy groups. However, the DARPA XG model does not support sophisticated meta-policies related to the authoring of policy rules. For this, a more sophisticated model of the policy subjects, or authorities in XG terminology, involved in defining policies is required, where groups of policy authors and the relationship between those groups can be represented as the targets of polices. Languages such as PONDER [damianou] allow meta-policies with policy authorship groups as targets, where these groups are defined in terms of organisation roles [lupu]. However, in the type of fluid organisation represented by CTVR, roles do not provide a sufficiently flexible means for modelling the organisation of authority in a way that allows policy authoring rules to be freely changed. The role-based model sees an organisation as a group of individuals and focuses on their positions and their relationship with the resources available to the organisation [sandhu]. As such there is often a mutual dependence between the semantics of different roles that means changes to one role causes changes to others in terms of their policy authoring capability. This stems from the tendency to attempt to use roles to model organisations in their entirety. We propose instead a community-based approach to modelling the targets of policy authoring meta-policies in that this better supports incomplete and fluid models of organisational structure. It is our belief that such flexibility will provide necessary if we are to support innovation in the organisational structures and spectrum access value chains that are described in section 1.

In the community-based system model, we view an organisation as a set of functional units, termed communities, which are linked to each other by means of a natural hierarchy of authority. The actions of authorisation meta-policy within the system are seen as being exerted on these units, i.e., the functional units are targets of such actions. Once an action is performed on a functional unit, the policy authoring properties of this unit may change or develop. The development of properties in a functional unit is described as "emergent properties". These properties are independent of individual actors in the system, i.e. the system users.

A key concept, in the community based policy model, is the idea of hierarchy. All the functional units in a system are placed within a hierarchy of authority. There are two main conditions that the hierarchy imposes on its members:

1. Membership - on each level of the hierarchy all the members of a particular community are members of the community directly above it in the authority hierarchy.
2. Authority – the authority of a community is superseded by the community directly above it in the hierarchy.

The hierarchy of authority emanates naturally from the idea that communities lower down the tree are members of the communities higher up the tree and therefore the goals of lower communities are to aid the higher communities to fulfil their goals. And so on, until the community at the root of the tree, "the organisation", has all subordinate communities striving towards the ultimate goal of the organisation as a whole. The organisation or root community has ultimate authority over all other communities and resources. The entire community hierarchy tree is itself considered a resource that can be managed with ultimate authority by the root community, and in a limited capacity, by communities lower down the tree. Communities higher up the organisational tree can issue a mandate to communities lower down the tree that delegate authority to a particular sub-community. On receipt of this mandate, the sub-community (e.g. an executive) can now control the organisation at this level of the tree and below. Mandates are issued by the root communities by means of policies, written by the root community, that allow the selected sub-community to act as an agent of the root community.

Figure 1 shows a sample hierarchy of functional units with in the organisation of the broadband spectrum users in the Irish republic; otherwise known as communities. COMREG has the authority to determine the area of the spectrum the CTVR has authority over. Although CTVR and COMREG are on the same level of the hierarchy tree, one appears to have authority over the other. This may seem to contradict the principle of a hierarchy of authority model. However, policies will have been written by "All Irish Spectrum users" that delegate authority to COMREG to determine the

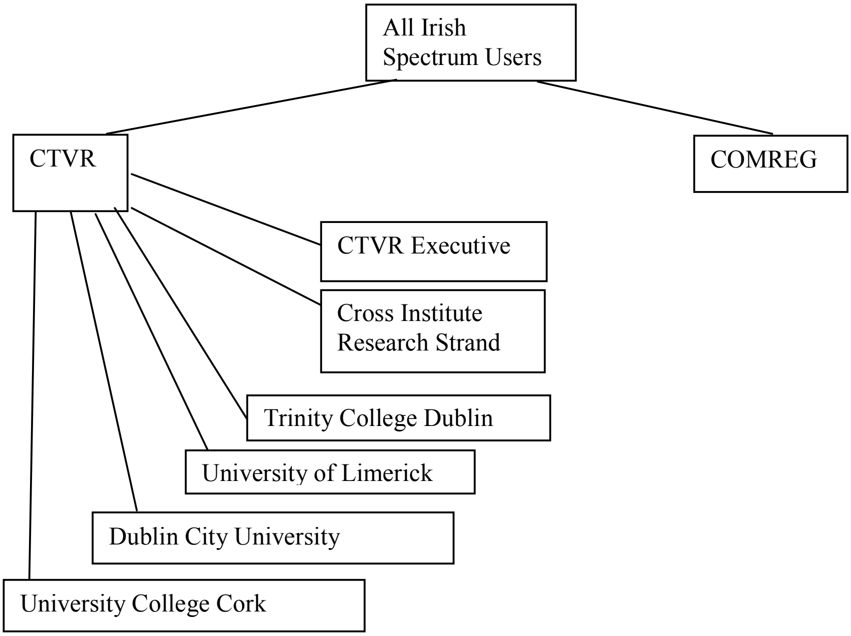


Fig. 1. Community-based policy management scheme for CTVR Software Radio test license

allocation of the spectrum to CTVR. Likewise, CTVR will write a policy to delegate authority of the spectrum allocation to the CTVR executive. In a scenario where the University of Limerick and Trinity College Dublin are running experiments that both requires the full use of the resource, the individual authorities they possess will preclude them from proceeding with the experiment without resolution of the conflict. The conflict is escalated up the tree until the node that has the authority over both competing communities is reached. In the case of this example this is the CTVR executive. The policies of CTVR executive are examined to reveal that the authority to administer full control the full spectrum has been delegated to the executive. The executive then makes a decision, based on its policies, which university should be allowed temporary full control of the resource, to run its experiment. The decision of the executive is passed upward to the CTVR group and then down to the spectrum users. This cycle is know as the “escalate-delegate” request cycle and is the main concept in conflict resolution. By allowing this cycle to be driven by organisational rule, this scheme move us towards the idea of an ‘autonomic organisation’, i.e. an organisation where conflicts are naturally identified and resolved through to day-to-day administrative structure.

However, the definition of hierarchical authority is insufficient to describe all forms of existing human organisation. There are many examples of human organisation where the hierarchy of authority of functional units with respect to resources does not hold – more general functional units do not always possess paramount authority over more specific ones. For example, a supply chain is a coordinated system of entities, activities, information and resources involved in moving a product or service from supplier to customer. The entities of a supply chain in dynamic spectrum access could consist of different forms of markets for exchanging commoditise portions of the spectrum under different sets of rules appropriate to device and frequency capabilities or application needs, e.g. terminal link or back-haul link in mesh networks. Supply chains are often formed by a collection of autonomous organisations, each with its own hierarchy of functional units and attendant hierarchy of authority. However, in the case of supply chains in dynamic spectrum access, it is frequently the case that the spectrum, as the primary resource in question, is not exclusively controlled by any of these organisations. Thus, it may be the case that none of the participating organisations has authority to take decisions relating to this resource without consulting the other partners – according to the agreement that forms the basis of the chain. Therefore, if we want to model the relationship between functional units and resources for the purpose of applying our community model, it is the community that represents the entire supply-chain that has authority over this database resource. The individual organisations that constitute the chain are now communities positioned beneath the community that represents the entire supply chain.

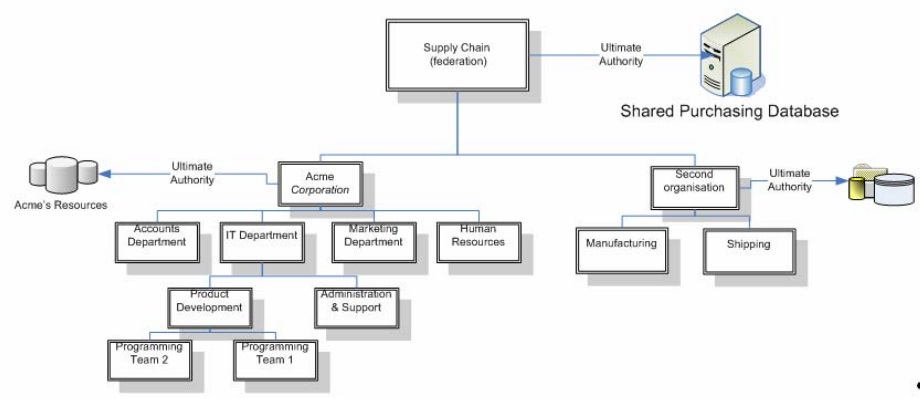


Fig. 2. Simple Schematic of Functional Units' Authority over Resources in Federation

In this case, the hierarchy of authority reflects our community model of the organisation only with respect to those resources which the overall supply chain possesses ultimate authority for. With respect to all other resources, including the resources representing the structures of the constituent organisations, ultimate authority remains with the communities representing the autonomous organisations that make up the supply chain. However, there is little problem in incorporating this type of organisational arrangement into our model. The hierarchy of authority still applies, but the hierarchy of authority for each resource is rooted in the community that possesses ultimate authority for the resource. Thus, the supply chain community does not possess any authority over the non-shared resources of the autonomous organisations that constitute it since ultimate authority for them is possessed by the community representing the organisation. It is only with respect to the shared resources that the top-level community is at the top of the hierarchy of authority.

Thus the location of the root community in the hierarchy of authority is dependant on the resource being acted upon and is equal to the community which possesses ultimate authority for that resource.

The type of organisational model described above in relation to the supply chain example, we refer to as a federation. It is distinguished from our definition of the organisation above by the fact that the root community possesses ultimate authority for only a subset of the resources managed by the overall community hierarchy. By incorporating the concept that ultimate authority for resources can be distributed among the communities in the hierarchy and by making the community with ultimate authority for a particular resource the root node in our hierarchy of authority for that resource, we can generalise our model to incorporate a very wide range of human organisational forms. The organisation described above – the traditional concept of the organisation – becomes a special case of an organisation, distinguished by the fact that ultimate authority for all resources controlled by the community structure is possessed by the top-level community. So, for example, if we were to consider that the various autonomous organisations involved in our example of the supply chain were to merge and adopt a traditional company form, we would merely need to move ultimate authority for all of the resources to the root community in order to model the change.

Once the community model has been defined, further enrichment of the model is required. Namely: the modelling of the resources available to the community. To model a resource in community based policy management, there is a requirement to separate a particular resource into two distinct sub-units: actions and targets. In the case of the example previous, the broadband spectrum is considered to be the resource. COMREG may wish to apply access control to that particular resource. There must be a target tree defined for the broadband spectrum. The target tree does not necessarily reflect the native hierarchical structure of the resource, but rather the access control requirements of the root community. Thus the tree for a spectrum resource could divide up spectrum by region or frequency or time or other application- or device- dependent capability, or some suitable combination of all of these to form spectral quanta thought suitable for commoditization of the spectrum. Also, as this is a hierarchy model may be possible for the course quanta defined by say a national regulator such as COMREG, to be autonomously subdivided in the finer grained quanta for the purposes of a secondary market.

Leading on from the idea of a target tree, is the action tree. Each resource available to the community has finite number of actions that may be performed on it. The action tree models these finite set of actions in a hierarchy. Therefore an entity that has authority to perform an action on a particular target also possesses the authority to perform any actions beneath the specified action on the action tree on the same target. With regard to the dynamic spectrum allocation model, the "action tree" is basic in structure. However in scenarios such as database administration, the action trees become far more complex and can span several levels.

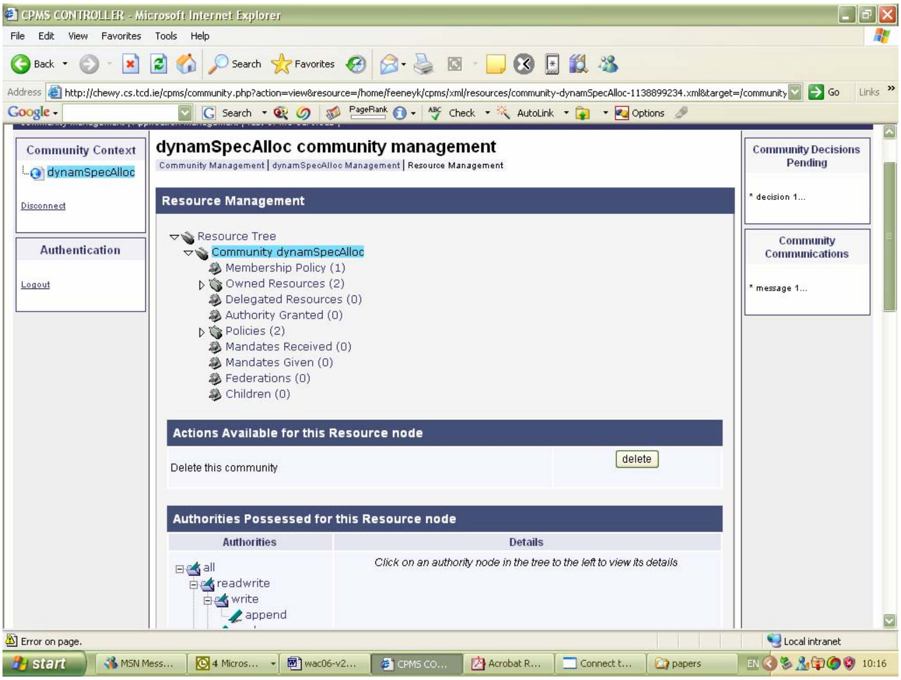
There is one final part of the community based policy management model that must be implemented to allow for the modelling of real world problems. The relationship of communities, resources and members must be more clearly defined. Thus far we have a model of the organisation in terms of its communities. Further, we have a model of the authority that these functional units possess over the resources of the organisation. We have now added a means of mapping the individuals that make up the organisation to these communities as members. However, we can not simply assume that each of the individual members can exercise the authority over the resources possessed by those communities that they are members of. For one thing, recall that in our model the root node of the hierarchical structure of the organisations is the community that represents the entire organisation and that this community possesses absolute authority over all of the resources controlled by the organisation. Further recall that each individual who belongs to the organisation is by definition a member of the root community. If each individual was able to exercise the authority of each community to which he belongs, each individual would possess ultimate authority over all of the organisation's resources through his membership of the root community and the community beneath the root would be obsolete in terms of informing access control decisions. Besides, it is rarely desired that every single member of an organisation should possess absolute authority over all of the resources controlled by the organisation!

Therefore, the final element of our model introduces the concept of a community policy set. Each community possesses a set of policies which dictate how the community's authority over resources can be acted upon by the members of that unit. In terms of our community model of the organisation, which considers the communities to be emergent entities with autonomous agency, we consider the community policy set as the decision making mechanism of the community. Often this would represent some form of collective decision making, e.g. majority voting in a committee style community, or more relevant to dynamic spectrum access, market rules (e.g. auction/bidding rules) for commercially based allocation, or fairness-oriented rules for open commons based access. Thus, when a community policy permits an individual member to exercise the authority of the community to act upon a resource, we say that she acts on behalf of that community and we consider her to be acting as an agent of the community which has approved her actions through its policies.

In summary, the community based policy management model consists of: defined functional units (communities) in a natural hierarchy of authority, a resource authority model that access control is required, a defined authority exercised by these communities on the various resources known as an assigned resource authority, a mapping of individual actors in the system to the communities that defines them as members of communities, and a policy set that defines the authorities possessed by actors in a community that those actors may exercise on behalf of that community. The combination of all these gives a rich dynamic model of an organisation or system that allows for easy distribution of authority as well as a powerful means of conflict resolution. Community based policy management is well suited to the specific problem of dynamic spectrum allocation and is highly given the extremely fluid nature of this problem.

4 Applying Community Based Policy Management to DARPA XG

The community based policy management scheme has been implemented as a web-based service using PHP, with a MySQL back end. This offers a comprehensive web interface for authoring community-based policies and meta-policies through a unified interface (see figure 4). This interface also reports on the detection of static (or modal) policy conflicts and identifies the point in the community hierarchy from where the conflict arose, and thus at which it must be ultimately resolved.


CPMS CONTROLLER - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://chewy.cs.tcd.ie/cpms/community.php?action=view&resource=/home/feeneyk/cpms/xml/resources/community-dynamSpecAlloc-1138899234.xml&target=/community
Go Links
Google Search PageRank Check AutoLink Options
Error on page.
Local intranet
start MSN Mess... 4 Micros... wac06-v2... CPMS CO... Acrobat R... Connect t... papers
EN 10:16

Community Context

dynamSpecAlloc

Disconnect

Authentication

Logout

dynamSpecAlloc community management

Community Management | dynamSpecAlloc Management | Resource Management

Resource Management

- Resource Tree
  - Community dynamSpecAlloc
    - Membership Policy (1)
    - Owned Resources (2)
    - Delegated Resources (0)
    - Authority Granted (0)
    - Policies (2)
    - Mandates Received (0)
    - Mandates Given (0)
    - Federations (0)
    - Children (0)

Actions Available for this Resource node

Delete this community	delete

Authorities Possessed for this Resource node

Authorities	Details
all readwrite write append	*Click on an authority node in the tree to the left to view its details*

Community Decisions Pending

* decision 1...

Community Communications

* message 1...

Fig. 3. Sample of the web interface to community

The server also offers the means to offer a policy decision service to other entities. This has been designed to offer a range of easy integration options. To date the policy decision service has been used to integrate the service with a PHP-based collaborative web portal for open source software development [feeney] via a direct PHP call and also to integrate with a decentralised context distribution service via an XML-based Pub-Sub interface. On this basis we are confident that community-based decision making could be readily integrated with implementations of DARPA XG policy decision points as they emerge.

Ultimately, however, our aim is to integrate the community-based policy concepts directly into the DARPA XG policy language using the extension mechanisms it provides. Such extensions should include the following:

- A means of defining a spectrum quanta as a resource. This could be based on existing XG descriptors for frequency, region, time and device descriptors, though anticipate an application descriptor will also be required.
- A means for defining rules for community membership, community mandates and community decision making mechanisms.

5 Conclusions and Future Work

In this paper we identify weaknesses in the DARPA XG policy language that prevent it being used to fully support flexibility and innovation in defining policies for dynamic spectrum access. Using even a simple model of a new software radio license we identify a lack of support in the language for modelling the organisational subject of policies, which in turn restricts the extend to which meta-policies can be used to define means, e.g. markets or fair open access, for sharing resources. We describe an existing policy-based management scheme based on the concept of communities and hierarchical authority delegation between them that can be used to provide such features in DARPA XG. In addition we show how the community based scheme can support the delegation of policy making authority in value chains, which regulatory trends indicate will become an important feature of dynamic spectrum access provision in the future.

Our future work will focus on using our existing community based management service for administration of the CTVR software radio licence. As part of this we aim to integrate community based policy management features as language extensions to DARPA XG.

This integration also throws up a couple of interesting research challenges in policy languages. Firstly, supporting ontology-based extensions to a policy language promises incremental improvement in it expressiveness, however it will be necessary to be able to handle the impact of extension on system with existing rules with elements that are effected by language extension, without the need of bring the system down and re-analysing all existing policies in the context of the extensions. For instance the formation of a new value chain between two dynamic spectrum brokerages may result in resources modelled by one becoming a subclass of a resource used in the other. In this case, all policies related to the latter resource suddenly also apply to the former, raising the possibility of a new set of unexpected policy conflicts. Tools for handling such conflicts without necessarily disabling the policy rules in question would be vital to the smooth operation of frequent value chains formation. A second research issue relates to the development of large policy sets. a key benefit of the community based scheme is that it allow policies to be incrementally introduced as the organisational model grows organically to meet the needs of the organisation or value chain. However, choices in seeding the structure of an organisation can have a strong impact on the efficiency of subsequent organisational change. Poor seeding may result in large numbers of conflicts results form each future change or conflicts that require further resource or action modelling to solve. Such outcomes can be regarded as negative metrics of the policy set so tools to predict the value of such metrics given a set of seed policy structures could provide invaluable. We are currently examining the potential of genetic algorithms and constraint programming techniques as a basis for such tools.

Acknowledgements

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

References

[damianou] Damianou, N., Dulay, N., Lupu, E., Sloman, M., (2001) "The Ponder Policy Specification Language" ,Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, 29-31 Jan. 2001, Springer-Verlag LNCS 1995, pp. 17-28

[feeney04] Feeney, K., Lewis, D., Wade, V. "Policy-based Management for Internet Communities", in proc of 5th IEEE International Workshop on Policies and Distributed Systems and Networks, IEEE, 2004

[kagal] Kagal, L., Finin, T., Joshi, A., "A Policy Language for A Pervasive Computing Environment", IEEE 4th International Workshop on Policies for Distributed Systems and Networks, June 04, 2003

[lupu] Lupu, E.C, Sloman, M. "Conflicts in Policy-Based Distributed Systems Management", IEEE Transactions on software engineering, vol. 25, no. 6, November 1999.

[mitola] J. Mitola III, G.Q. Maguire Jr, "Cognative Radio: Making Software Radios more Personal", IEEE Personal Communications Magazine, vol 6, pp 13-18, August 1999

[owl] World Wide Web Consortium (W3C), Web Ontology Language (OWL), www.w3.org/2004/OWL/, Visited Mar 2005

[sandhu] Sandhu, R.S. et al., "Role Based Access Control Models", IEEE Computer, vol. 29, no.2, 1996, p 38-47

[uszok] Uszok, A., et al. "KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement" Proceedings of IEEE 4th International Workshop on Policies for distributed Systems and Networks, June 2003 Lake Como, Italy pp 93-99

[xg-policy] DARPA XG Working Group "DARPA XG Policy Language Framework, Request for Comments", version 1.0, prepared by BBN Technologies, Cambridge MA, USA, April 16 2004

[xg-vision] DAPRA XG Working Group, "The XG Vision Request for Comments". Version 2.0, prepared by BBN Technologies, Cambridge MA, USA, January 2004

An Intermediate Framework for Unifying and Automating Mobile Communication Systems

G. Koumoutsos, K. Lampropoulos, N. Efthymiopoulos, A. Christakidis, S. Denazis, and K. Thramboulidis

University of Patras
{koumouts, klamprop, nefthymiop, schristakidis, sdena, thrambo}@ee.upatras.gr

Abstract. In the last few years we are witnessing a rapid growth in the mobile use of the internet and services like telephony. Mobility offers a great deal of advantages to end users but has also restrictions due to heterogeneity of networks, protocols and mobile devices with limited capabilities. In that we can add the need for manual interaction with service providers which costs in time and flexibility. We propose a framework that solves major mobility issues like connection maintenance, roaming, automated service selection regardless the underlying devices, protocols, providers and services. We describe important steps that will automate procedures and became the bases for an autonomic mobile environment transparent to the end user.

Keywords: Mobile Router, SLA, negotiation, matchmaking, ontologies, network mobility, addressing, roaming.

1 Introduction

The rapid growth of the internet resulted in the proliferation of services and technologies comprising a complex network mosaic of various access methods, protocols, frequency bands, data speeds, and core transport networks. Different administrative domains and ownership add to this complexity through a variety of authentication, billing schemes, domain services and applications creating a highly heterogeneous environment, which is called to sustain a “homogeneous” end-to-end path as perceived and experienced by end users.

In contrast, users’ ways of working and interacting have also drastically changed as they have become more mobile and dependent on the internet. Their primary, albeit elusive, requirement is ubiquitous connectivity through a wide range of devices and maintaining the same user environment/experience.

Unfortunately, users today are not in a position to recognize services or applications as distinct commodities where they can be purchased or negotiated from interacting with the network, as the only recognizable service today is mere connectivity with a specific provider or providers. Even this type of rudimentary service requires users’ intervention and time in order to negotiate and sign a contract with the provider which in turn restricts the user to function within the limitations of this specific provider. Moving to an area of non coverage entails new negotiations and contracts (WLAN access) and/or higher fees (roaming).

D. Gaiti et al. (Eds.): AN 2006, LNCS 4195, pp. 298 – 314, 2006.

The aforementioned issues have been our motivation behind our vision described in this paper according to which, users are capable of engaging themselves in a higher form of "conversation" with the network shopping for services without even having the knowledge of who the provider(s) is. Services and a "language" for requesting them are the main entities that the user recognizes and can access anytime, anywhere and most importantly by any available means or device. To this end, the device and the technology take a backseat – they are just the means for provision, and services become the true commodity on offer, thus, giving real meaning to globalization of networking and computing.

Driven by this vision we have identified a number of steps and issues that require immediate attention and solutions.

2 Towards Autonomic Communications

Our vision is to create a service-centric, user-friendly environment where a mobile-end-user will have the ability to move through different networks, providers and acquire services transparently, letting machines deal with the advanced complexity issues. Until today a mobile user has to manually:

- choose provider and specific service
- buy a connection service like any other product
- configure his devices for connection
- monitor service performance and act accordingly

So he is restrained by:

- predefined and non-adjustable specific provider and service terms selection
- providers' network coverage
- complex and time consuming procedures (configuration, services purchase) which require expert's knowledge

In our scenario the same user is moving across town using a device (PDA, Laptop, mobile phone, etc …) to connect to the internet. Using our intermediate framework he has the ability to:

- automatically search for accessible networks, providers and services
- choose and acquire the appropriate service based on his demands
- monitor the quality of the service
- dynamically change network, provider, service or user end-device and maintain connection
- make use of certain advice capabilities to help with essential knowledge needed

3 Architecture

Our framework is based on a mobile prototype device called Chameleon Router (CR). This device will stand at the frontiers of end-users world, adopting to environments and automating procedures (Figure 1). It is a device with all basic hardware characteristics and a number of slots that can accept different kinds of network cards (NIC), interconnecting the mobile users devices (personal area network -PAN) with wide area networks (WAN).

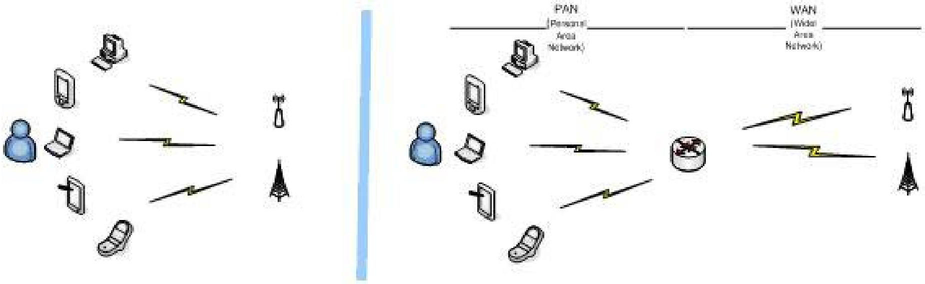


Fig. 1. Mobile connectivity without and with our intermediate framework

On boot, CR performs a scan in multiple interfaces for all types of available networks in both sides: PAN and WAN. After making a list with the available choices, it begins the service selection procedure in order to select for his user the appropriate service. A negotiation scheme is launched depending on user's predefined preferences with all available service providers. Matchmaking takes as inputs user's requirements and available network-provider-service choices to select the best matching service. Then, an SLA (Service Level Agreement) is "signed" between two parties (user and provider) and the use of the service begins. From that point, a monitoring agent observes the consistency of the service based on the SLA, notifying user for any abnormal behavior.

3.1 Acquiring Services

In the direction of automated service discovery, negotiation, selection, SLA binding and monitoring multiple modern tools, protocols and technologies can aid. In order to automate these procedures, "machines" (mainly software intelligent agents) have to be given the means to interoperate, understand each other, translate the well described knowledge and make decisions in account of their owners. In our scenario we will examine closely:

1. Pre - configuration of our agent according to user preferences.
2. Service discovery on user's-application's request.
3. Negotiation and service selection.
4. Matchmaking and SLA forming.
5. Monitoring and reaction in any case that can arise.

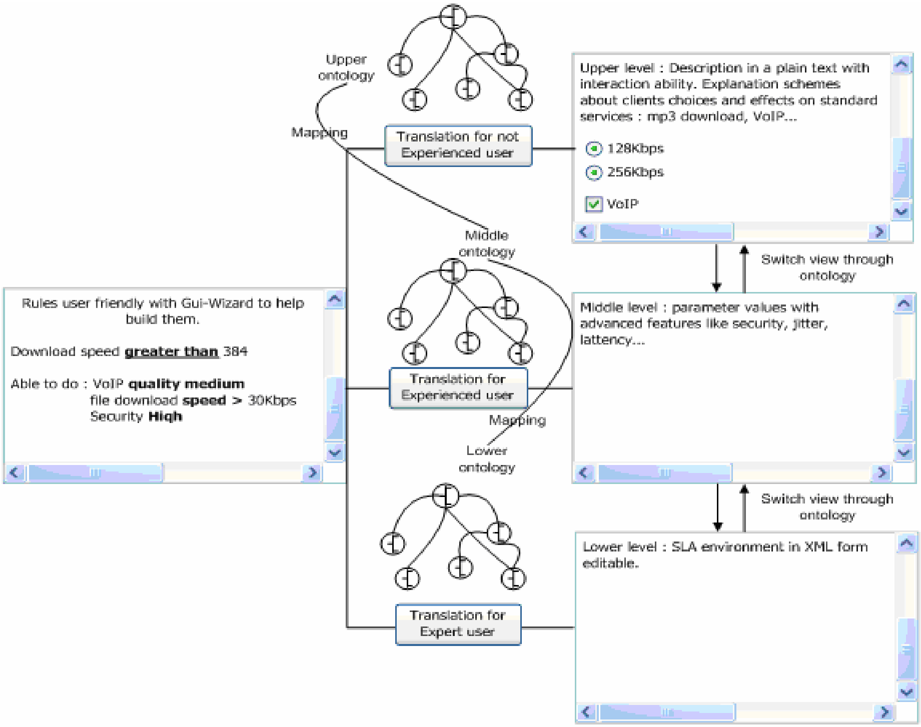
In order to achieve all the above our device will include significant processing and memory capacity, as well as wide area wireless access, so parts of application code, as well as infrastructure functions such as packet forwarding, routing, QoS support and service selection, will run on the device itself. It will have an interaction GUI that will use model-driven-architecture techniques user friendly in order to be preconfigured at the beginning and altered every time that is needed. Our approach doesn't use AI in service discovery, matchmaking and selection as in MIT's suggestion [1] where a system is educated to model users behaviour and make the most appropriate choices. Instead we will use mostly modern semantic tools to describe and make machines understand knowledge in order to discover it, standard negotiation schemes and protocols that will use flexible rules and ontologies that can easily be customised and adapted, parameterized matchmaking algorithms that will -with the pre-configuration of the user- make the best choice. We will use and extend all knowledge from recently very developed Semantic Web Services (SWS) world to achieve automation and transparency to the end user. SWS provide an approach for representing the functionality of Web services with the help of ontologies. Popular approaches for SWS include OWL-S[2], WSMO[3], FLOWS[4] and WSDL-S[5]. In service level we think of our framework as a special case of a web service as we have to contact a provider, understand the "service" he is offering, negotiate and choose based on our needs. We won't have to use all tools and protocols (e.g. SOAP) from web services but we can take advantage of semantic annotation with ontologies and rules, agent cooperation, interaction schemes and other. Our main goal is to adjust all this tools to the mobile agent as described with all limitations mobility and devices could put in our way. Moreover in that way a web services enabled agent will arise ready to absorb and take advantage of this fast evolving area. We are planning to bring as many capabilities as possible on client's side giving a powerful agent that will automate procedures to the end-user.

3.1.1 User Interaction

Our agent will be accessible through a simple user interface that will be adjustable to each user's device. It will be the tool to create a profile of the service the user will need in various cases. Through it he will have the ability to form rules that will create the desired service based on his knowledge. It will have multiple interaction schemes and description-aspects in order to let naïve and expert users configure their system just as well. In figure 2 we can see how the user will be able to create his desired service profile through ticks, clicks and wizards, at the left window, and how depending on his choice he will see the result in three levels of understanding. At the top level he will see his choices in a plain very simple text with explanations, choices effects on standard services and advises based on systems knowledge as we will explain later. The next two levels will include advanced features in order to let more experienced users configure their system more precisely. The user will have the ability to edit in both sides based on his experience, see his choices from multiple aspects and configure his system. All this aspects of the same description of a service will derive from an ontology that will describe the service area and map between different level-views.

It will be sub-ontology of the SLA-ontology which will have a lot more in order to fully describe the relationship between the two parties. Except from this aspect-oriented service ontology, rules are going to be used for multiple reasons:

1. Rules to extract information from provisional SLAs and calculate effects on standard services along with explaining what each parameter is to naïve users. This will be part of the advice capabilities of our framework that will help users as they build their best-suit service profile as well as during the selection process if needed.



Upper level : Description in a plain text with interaction ability. Explanation schemes about clients choices and effects on standard services : mp3 download, VoIP...

● 128Kbps

● 256Kbps

☑ VoIP

Middle level : parameter values with advanced features like security, jitter, lattency...

Lower level : SLA environment in XML form editable.

Rules user friendly with Gui-Wizard to help build them.

Download speed **greater than** 384

Able to do : VoIP **quality medium**
file download **speed** > 30Kbps
Security **High**

Fig. 2. Users profile creation ontology-based interface

2. Matchmaking decision rules with the aid of ontologies that will enable a much more accurate approach than the syntactic one available on today's XML based transactions.
3. Editable preference rules that will give the user the ability to customize his agent adding information to the existing ontology based profile. He will be in position to extend existing rules and create new that fit him better. This will be done through a user-friendly model-driven based interface described above.

Writing rules precisely is difficult even for logic experts. The rule editor let users write facts and rules as if they were writing natural sentences. The user simply selects a predicate for a fact or rule as a constraint on a parameter and sees the result on the right side in one of the three described views.

After the user has described his provisional SLA the agent forms it according to the ontology so it can be sent and understood by ISP's intelligent agents.

3.1.2 Scan

One fundamental issue we have to deal with, in order to succeed a reliable communication base is first of all the good knowledge of the available surrounding networks. Chameleon design has a number of slots where many different network cards (NIC) can be adjusted dynamically. The end user can insert at any time a new network card in a free slot of chameleon. The activation and configuration of the new card will automatically be managed by hotplug. This means that it will be functional immediately without the need of a device reset. So with a synchronous “all interfaces scan” the system may acknowledge new detected networks or keep track of much information about already existing. Such information is the link level, the power transmit, etc. Afterwards, the measurements are processed and a decision is made, whether there is an anomaly in the connection and if a transition is considerable, to a new more stable network.

An additional feature of the proposed system is the ability to identify whether specific QoS protocols are available e.g. 802.11e and trigger relative features in the NIC.

3.1.3 Discovery and Negotiation

The discovery of services follows the initial scan for networks. At this point we can directly “ask” the providers that were scanned for services or a special repository such us UDDI for services can be questioned. In direct contact with the provider multiple negotiation schemes can be adapted from FIPA [6] (Foundation for Physical Agents) where formal definitions of several standard negotiation protocols are presented. Another approach is that of WS-Agreement [7] a standard proposed by Global Grid Forum [8]. Depending on our approach this choices can be described using semantic tools so it can be understood by both parties and dynamically adapted based on users preference or preconfigured strategies in the agent. For example our client after using his PDA during his transportation he boots his powerful laptop in a friends house. Our agent scans the change in its owner’s environment and automatically –or after asking based on its pre-configuration- searches for a more appropriate service based on new needs that may arise. The new environment states a more stationary condition and more time of possible use of the service. This means that a more persistent search and a possible exhaustive auction is more appropriate instead of a simple request-replay that it used during the transportation and the PDA use. The description of this multiple negotiation schemes with semantic tools will make them easy to understand and be used. In more advanced scenarios this schemes will be easy, through model-driven techniques, to be altered by simple users and providers that want to design their own strategies in negotiations. In that way we will avoid hard-coding one negotiation protocol to our agent, which must be known in advanced by all parties. Instead we will design a basic interaction protocol capable of supporting all possible negotiation schemes that will be described. As we see in figure 3 such a protocol will act as a platform capable of understanding descriptive rules or transaction ontologies and adapting. It will have only hard coded the information needed to form lower communication levels and understand the semantic guidance.

The idea of dynamically adapting multiple negotiation schemes has been proposed with multiple variations. Bartolini et al [9] defines an agent-based generalized interaction protocol that can be specialised with rules very similar to our approach. They provide a taxonomy of such rules for various negotiation mechanisms with good results for a general negotiation protocol that will be in position to deal with everything. Tamma et.al [10] also describes an approach to negotiation, where the negotiation protocol does not need to be hard-coded in agents, but it is represented by an ontology: an explicit and declarative representation of the negotiation protocol. In this approach, agents need very little prior knowledge of the protocol, and acquire this knowledge directly from the marketplace. The ontology is also used to tune agents' strategies to the specific protocol used.

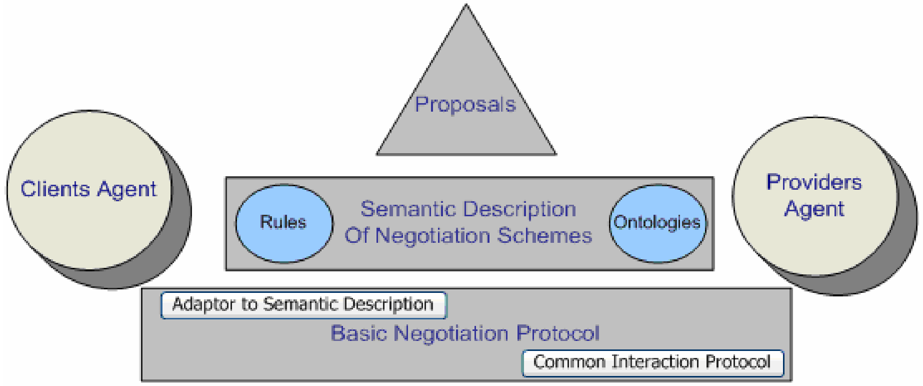

Proposals
Clients Agent
Rules
Semantic Description Of Negotiation Schemes
Ontologies
Providers Agent
Adaptor to Semantic Description
Basic Negotiation Protocol
Common Interaction Protocol

Fig. 3. Adaptable negotiation stack

We intend to use ontologies to describe the concepts and relationships that are shared across major negotiation protocols in a higher level negotiation ontology. This will provide us with the basic vocabulary that the negotiation agents must share. Upon this and with the use of other domain specific ontologies like time and FSM (Finite State Machine) ontologies we can describe multiple negotiation schemes. An engine capable of understanding and using this basic ontologies is going to be implemented with the ability to adjust dynamically to described schemes. This will give us the ability to describe new strategies based on the same basic ontologies as described above.

In figure 4 we can see a negotiation scheme which describes a "small" auction that gives the opportunity to service providers to improve their suggestion only ones before the final decision is made. The first step for the end user is to create his service profiles with the aid of above described ontology based interface. We use "profiles" because the client will be able to design multiple service schemes in order to use it in different circumstances, with different end devices. Chameleon receives both profiles and providers list after scanning. It then produces and sends client's goals-provisional SLA- to providers. The matchmaking process takes place after receiving provider's prepositions two times before the final decision is made. The UML diagram will be formed in a transitional ontology based on the basic ontologies which our basic hard-coded engine and protocol will understand and enforce. We can thing multiple strategies that can be developed based on available negotiation schemes and our vision is to unleash this potential with semantically described - easy adopted schemes.

3.1.4 Matchmaking

Our model of software-agent communication is based on the assumption that two agents, who wish to converse, share a common Service ontology which ensures that

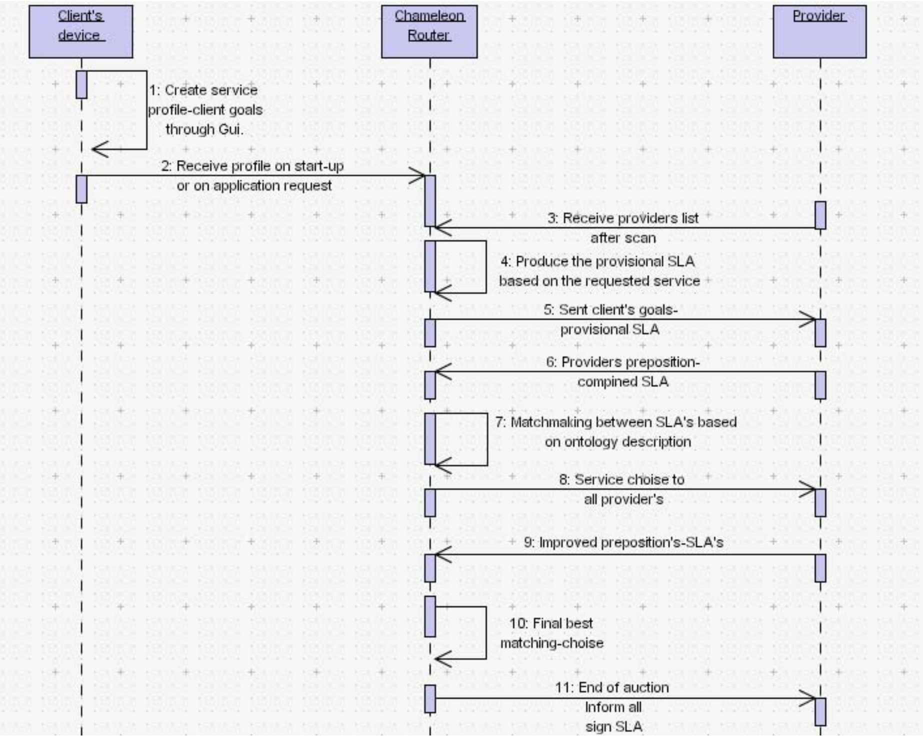

Client's device
Chameleon Router
Provider
1: Create service profile-client goals through Gui.
2: Receive profile on start-up or on application request
3: Receive providers list after scan
4: Produce the provisional SLA based on the requested service
5: Sent client's goals-provisional SLA
6: Providers preposition-compined SLA
7: Matchmaking between SLA's based on ontology description
8: Service choise to all provider's
9: Improved preposition's-SLA's
10: Final best matching-choise
11: End of auction Inform all sign SLA

Fig. 4. A possible negotiation scheme in a transition diagram

they ascribe the same meaning to the terminology used. In an open environment, agents are designed around various ontologies (either implicit or explicit). For allowing their communication, explicit ontologies are however necessary, together with a standard mechanism to access and refer to them (such as an access protocol or a naming space). Without explicit ontologies, agents need to share intrinsically the same ontology to be able to communicate and this is a strong constraint in an open environment where agents, designed by different programmers or organizations, may enter into communication. Explicit ontologies can be considered as “a referring knowledge” and, as a consequence, can be outside the communicating agents. FIPA00006 specification deals with technologies enabling agents to manage explicit, declaratively represented ontologies. We intent to build upon this specification, using modern techniques and tools. In figure 5 we can see a diagram showing the two sides –clients and providers- along with the basic function blocks of our framework. ISPs form their provisional SLAs based on multiple ontologies that can be easily located in a public repository. They can also create their own ontologies as long as they give a way for clients to interact with them. On the other side clients create their service profiles according to their multiple aspect ontology as described above in the user interface.

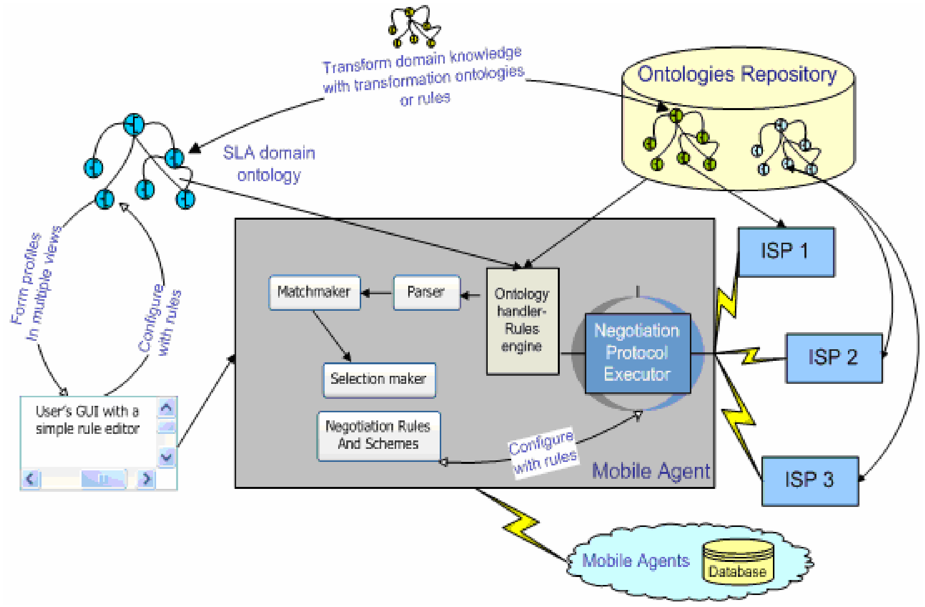

Transform domain knowledge with transformation ontologies or rules
Ontologies Repository
SLA domain ontology
Form profiles In multiple views
Configure with rules
Matchmaker
Parser
Ontology handler-Rules engine
Selection maker
Negotiation Protocol Executor
Negotiation Rules And Schemes
Configure with rules
Mobile Agent
User's GUI with a simple rule editor
ISP 1
ISP 2
ISP 3
Mobile Agents
Database

Fig. 5. Service acquiring architecture with basic components

The way to make the two worlds understand each other and interoperate is to translate service domain knowledge either with transformation ontologies or with rules. Both seem to be very powerful and promising tools in semantic annotation and reasoning, and currently there is ongoing work on combining both. Along with their power, questions regarding the need for such rich languages due to several complexity and computability barriers have aroused.

Our goal is to examine how this powerful but expensive, in terms of complexity, tools can fit in the more limited mobile world as part of our agent. Our internet service specific domain environment could be possibly solved with easier static approaches but in that way we would not create a mobile agent ready for the full semantically automated tomorrows web.

Our service selection mechanism need to be based not only on semantic service description given from the perspective of providers but also consider pragmatics which builds on but is deeper than semantics [11]. For that we intend to give our agent the means to cooperate with other agents on evaluating service providers and contacting special repositories (databases) that keep useful information for the selection process.

3.1.5 SLAs

An SLA specifies agreements between a service provider and a customer and the measurements to be taken in case of deviation and failure. The domain of SLAs is recently very developed as many companies and ISPs are developing tools to represent the relationship between two cooperating parties. Again in web services area we have WSDL (Web Services Description Language) which despite the name does not provide a means to express services capabilities; therefore such standards as WS-Policy [12] WSLA [13] and WS-Agreement specification exist to allow for the expression of additional attributes in order to discover, match and use a web service correctly. All the above standards will be our guide in forming an appropriate SLA ontology that will include service ontology, signatory and supporting parties as well as obligations and actions to be taken [14].

In a recent works [15] semantic tools (owl, ontologies, rules) are used able to reason over ontologies and retrieve data via ontology queries with much less effort giving more accurate matches. This work focuses on matching Service Level Objectives in the WS-Agreement specification with reasoning over qualifying conditions and business values. It combines OWL ontologies of the specific domain, QoS, agreements and ARL rules to provide the matcher with detailed knowledge. Our work is based on the same principles and technologies but deals with the more specific area of internet service in a mobile agent a restriction that will not let us use already developed tools as in a stable powerful machine. All described procedures have to adapt in the mobile environment keeping close all basic functionality and putting on remote sites everything that can be remotely invoked. On a web services ready agent remote invoking will not be a problem and that is the reason we designed our framework based on SWS.

3.1.6 Monitoring SLAs

Monitoring and management of SLAs is an important aspect of the relationship between the two parties and it is described in the SLA ontology. There we will have a complete description of the obligations of two signatory parties which will specify service parameters to be monitored with metrics and functions. It will allow other parties involve in the monitoring phase with details on their exact role as well as describe actions to be taken in any case that can arise. It will be like any other contract between companies, in the domain of service, properly designed to include all the above in order to automate monitoring and reaction. IBM's WSLA gives a description of such an SLA in an static XML description. We intend to build upon it a dynamic ontology model with all advantages this will mean for our framework.

3.2 Network Mobility

One major issue to be solved is the continuous connection to the Internet. A mobile user wants to move between networks like 802.11, 3G, etc and communicate with hosts using the internet, or by creating ad-hoc networks. He also wants to have the ability to choose his favorite provider, and have continuous connection from any network to any network. But is the mobility problem a standalone problem or is it only a part of a huger issue, which is arising nowadays. Mobility is maybe the first negative sign of the tension of different services and networks to work under one communication system.

For example, Wi-Fi networks either using access points of ad-hoc schemes provide wireless connections to wireless users, and can be gateways to the internet. The addressing scheme serves only the mobility within one access point or, if TAP DANCE [16] is available, between access points of the same provider. To be more specific, moving to a different provider presumes the change of the end user's IP, something that causes all active connections to fail.

GSM and 3G mobile networks today provide internet services through mobile telephones, via WAP, GPRS e.g. The transition, though, from one provider to another is not yet available and the end user's choices are limited to the services and the network availability of his initial selection. Still it is obvious that even if one can use different providers, the capability for a smooth transition is impossible due to addressing problems.

Telephony as well is another major technology that is adjusting to the internet the later years. VoIP is already in use, as it is designed based on the internet infrastructure. On the other hand as far as it concerns the traditional telephony networks there is an effort from an IETF Working Group for E.164 [17] numbers to be expressed as a Fully Qualified Domain Name in a specific Internet Infrastructure domain defined for this purpose (e164.arpa).

Yet numerous other kinds of networks exist, like peer to peer, vpn's, etc. All these use specific protocols, addressing and routing schemes and operate over the internet using existing technologies only as the medium.

It is obvious that the combination of many different types of networks increases not only the complexity of the mobility problem, but generally creates many communication problems due to different addressing schemes, address translations and routing operations. Accordingly, we argue that, a global and scalable mechanism that unifies addressing in a way that is independent of networks, and addressing formats is critical for achieving ubiquitous and seamless connectivity.

We propose a new mechanism that unifies network addressing and works on the existing internet infrastructure based on peer to peer network techniques aiming at finding any host on the internet.

Similar effort towards the solution in a unified network can be found in "Internet Indirection Infrastructure" (i3) [18]. The idea of i3 contains the nodes and a hardware infrastructure of a system based on CHORD's [19] network model. Every node registers its id in the infrastructure. So when someone is trying to communicate with that node, searches in the overlay for the intermediate point where the node is registered, and forwards the packets to it. The intermediate point is then responsible to deliver the packets to the final destination.

In contrast we suggest a scheme for a fully distributed overlay naming bind system, based on CAN P2P network model [20]. The idea is to create a distributed overlay network on a virtual space, and bind a fixed point with a variable value like the physical address. In that space, all nodes will use a unique and by that, each one exists only in a specific point. Anyone who wants to communicate with a node, all he has to do is to look in the overlay for the specific id, and ask the node for its current IP address. So instead of having intermediate nodes forwarding our packets, we have nodes which discover each other only once, and then communicate based on the existing IP technology. We could compare the procedure with a distributed DNS model which is continuously updated for every IP of its nodes. This way, our system can be scalable and have better performance because data travel shorter distances towards their destination. QoS is not affected and the recovery of a loss node is not invidious for our system because the time to find the node once is much very small.

CAN is a structured P2P network which creates a d-dimensional virtual space. At every node that participates in the overlay is assigned a portion of that space (which is called zone). When a node wants to enter the overlay it hashes its IP address. The product of that hash function is some coordinates that correspond to a point in the virtual space. The node, then, contacts the node that is responsible for that point in the overlay, splits that node's zone in a given dimension and takes the one of the two new zones that were produced.

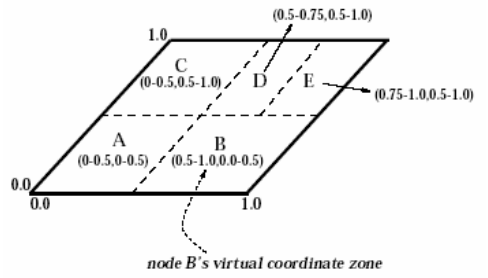


Fig. 6. CAN space distribution [20]

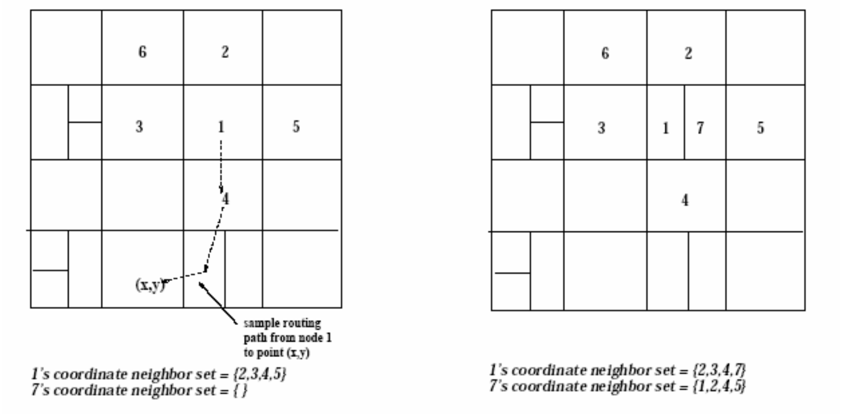

6
2
3
1
5
4
(x,y)
sample routing path from node 1 to point (x,y)
1's coordinate neighbor set = {2,3,4,5}
7's coordinate neighbor set = { }
6
2
3
1
7
5
4
1's coordinate neighbor set = {2,3,4,7}
7's coordinate neighbor set = {1,2,4,5}

Fig. 7. CAN node insertion [20]

In figure 6 we can see an example for a 2-d space with 5 nodes and the space distribution. Moreover in figure 7 it is shown an example with the insertion of a new node (node 7) and the re-distribution of the space in the overlay. Every node maintains a structure, called routing table, with the nodes (called neighbors) that have zones adjacent to its zone. This structure has the IP addresses of its neighbors and the coordinates of their zones. With the help of that structure any node can contact every other node in $O((d/2)(n^{1/d}))$ hops (where d is the dimension of the overlay and n the number of the nodes) by forwarding a message in every hop closer to its destination.

Our system is based in CAN but differs in two ways. First, nodes enter the overlay not according to the hash product of their network address but according to the hash product of a unique id, H(id), in exactly the same way as i3. The unique id could for example be a URL. So, now the point in which a node is mapped in the overlay is the same every time the node participates in the overlay and independent of its network address. Secondly, when a node N enters the overlay it contacts the node who is responsible for the point $H(id_N)$ and splits its zone not in the middle but in a way such as each node i has in its zone the point $H(id_i)$.

When a node wants to contact another node with id=x it routes through the overlay to the point H(x). If the node which is responsible for that point has id=x it sends back its network address and the addresses of its neighbors, otherwise it responds that the node with id=x is not in the system. In that way any node can discover if any other node is in the system, if it is can learn its network address and can establish a direct connection with it.

In the case of a mobile user which changes his network address the only thing that needs to be done is that the mobile user must inform its neighbors about its new network address and nothing else (his place in the overlay remains the same as that is independent of its network address). If that happens during a transaction with another node N, that node detects a timeout and contacts the neighbors of the mobile user, which their addresses has previously cached, to obtain its new network address. If the cached addresses are stale, which means that those nodes have also changed their addresses, node N uses again the overlay to obtain the new address of the mobile user.

Our system inherits from the CAN overlay its stability, scalability and fault-tolerance even if it is comprised of unreliable components. It is fully distributed and doesn't need any special infrastructure as in i3. Every node that participates in the system can discover any other node, if it knows its id, in $O((d/2)(n^{1/d}))$ hops and, most importantly, can make a direct connection with it. There is no notion of redirection as in i3 and mobile IP architectures. The recovery process, when a node changes its address during a connection, is very fast and with high probability O(1).

We decided to use the CAN overlay instead of any other structured DHT system, like Chord or Pastry [21], for its consistency mechanism. Unlike the other DHT systems, in CAN every node knows which nodes will be affected by a change in its network address and informs them instantly. In Chord, for example a change in the network address of a node will result in stale entries in the routing tables of other nodes with no easy way to be updated instantly. Additionally, CAN is flexible by the perspective of the number of dimensions used to define the virtual space. The number of dimension is a tradeoff between the number of nodes, desirable consistency level and bandwidth overhead for maintenance.

4 Related Work

Agent negotiation research has focused on the specification of protocols, often using conversations [22] specified as finite state machines. FIPA and WS-Agreement has defined various interaction protocols, including English and Dutch auctions [23]. Pitt [24] defines a semantic framework around FIPA ACL to allow the easier specification of multi-party interactions by adding structured conversation identifiers and a richer representation of protocol states. Multiple other efforts have been made like server-based auction rules and contract templates [25], the more general [26] which defines a formal approach for electronic agents interaction, [27] which covered the auction design space classifying mechanisms according to several parameters. Our approach takes the above as inputs and goes beyond defining a library of protocols, to a dynamically adaptable basic negotiation protocol that can be parameterized through semantic annotation.

In the domain of Service Level Agreement (SLA) major work in negotiating and matching is purely syntactic. Yang [14] developed a methodology for matching Web Service Level Agreements (WSLA). There SLAs are syntactically matched by parsing them into syntax trees and comparing them node by node. Heterogeneous SLAs are handled by referencing a table containing instructions to convert them into the same format. Such syntactic approaches must take a more exhaustive and laborious approach to matchmaking and are challenged by less obvious matches. In Paschke [28] a rule based SLA language (RBSLA) is used to express Service Level Agreements. The rules are based on the logic components of Derivation, Event Condition, Event Calculus, Courteous Logic, Deontic Logic, and Description Logic. Rule based SLAs can be written and modified using the management tool (RBSLM) which also enables the management, maintenance and monitoring of contract rules. Uszok [29] has developed KAOS using Semantic Web technologies for the specification, management, analysis, and enforcement of policies. The policy is represented using concepts from an OWL ontology. Our work also builds upon semantic tools and adjusts them in the mobile environment.

Until today many solutions for the mobility problem have been presented in relative projects with mobile routers, with the majority of them based on the Mobile IP [30] protocol. All of them try to approach the mobility problem using existing protocols and technologies and in most cases introduce solutions for specific network models, adopting the Mobile IP scheme for the rest of them. The IOTA (Integration Of Two Access technologies) [31] project with a network element called *IOTA gateway* it provides seamless connectivity for a user with AAA authentication services in different networks, but solves the mobility issue by implementing Mobile-IP agents. MAR (Mobile Access Router) [32] project introduces a mobile router for public transportation which provides to the end user aggregated bandwidth of multiple different connections, and shift load poor quality better quality channels. But it doesn't provide mobility between the different interfaces, in case one of them is disconnected. MPA (Mobile People Architecture) [33] project is focused on person-to-person reachability with a connection between two end users and not between a user and the internet. Personal Router [1], a project from MIT is focused only in developing an intelligent system that takes decisions about the choice of the available networks, according to the usual preferences of the mobile user, and selects a network depending on the accounting and QoS preferences of the user.

Only MobileNAT and Mobile IP proposals introduce schemes for mobility, but the first requires hardware Anchor Nodes in essential spots in the internet, and the second requires software Agents to redirect the traffic. It is clear that scalability and additional data load is analogical to the number of mobile users.

5 Future Work

We proposed a complete framework that will automate procedures and adjust useful tools in the mobility area. In order to make our scenario work we have to deal with constraints mobility puts in our way. In service selection a lot of work has be done in all involving procedures (negotiation, matchmaking, SLA formation, monitoring and management). The problem is that this suggestions are not based in a mobility environment and thus most of our work is to experiment on how this are going to be adjusted and how remote invocation can lighten our mobile agent. As far as it concerns the mobility proposed scheme, future work will focus on simulations and measurements to check if and how much the balance of the system if affected by the way nodes enter the overlay, and check potential bottlenecks due to the heterogeneity of the participating nodes and networks.

Acknowledgments

This work is funded by the Greek General Secretariat for Research and Technology in the context of PENED 2003 03ED723 project, (75% EC, 25% Greek Republic, according to 8.3, 3rd Framework programme).

References

[1] D. Clark and J. Wroclawski, "The Personal Router Whitepaper", Available online: http://www.ana.lcs.mit.edu/papers/PDF/PR_whitepaper_v2.pdf

[2] OWL-S, OWL-based Web Service Ontology web site, http://www.daml.org/services/owl-s/

[3] Web Services Modeling Ontology web site, http://www.wsmo.org

[4] SWSF web site, http://www.w3.org/Submission/SWSF/

[5] WSDL-S web site, http://www.w3.org/Submission/WSDL-S/

[6] The Foundation for Physical Intelligent Agents web site, http://www.fipa.org

[7] Andrieux, A., Czajkowski, C., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M., "WebServices Agreement Specification (WS-Agreement)", Available online: http://www.cct.lsu.edu/personal/maclaren/GridPrimer/Sources/WS-Agreement Specification.pdf

[8] Global Grid Forum web site, http://www.ggf.org

[9] Bartolini C., Preist C., Jennings N.R.: Architecting for Reuse: A Software Framework for Automated Negotiation, in Giunchiglia, F., Odell, J., Weiss G. (Eds.): Agent-Oriented Software Engineering III (2002), Springer-Verlag LNCS 2585/2003.

[10] Valentina Tamma , Steve Phelpsa, Ian Dickinsonb, Michael Wooldridgea, "Ontologies for supporting negotiation in e-commerce", Engineering Applications of Artificial Intelligence, 18 (2005) pp. 223–236, Elsevier

[11] R. Sreenath, M. Singh, "Agent-Based Service Selection", Journal of Web Semantics (2004)

[12] The Web Service Policy Framework, Available on line: http://www-106.ibm.com/developerworkds/library/ws-polfram

[13] The WSLA Specification Available online, http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

[14] W. Yang, H. Ludwig, A. Dan, “Compatibility Analysis of WSLA Service Level Objectives”, Workshop on the Design of Self-Managing Systems, RC22800 (W0305-082) Computer Science, 2003

[15] N. Oldham, K. Verma, A. Sheth, F. Hakimpour, “Semantic WS-Agreements Partner Selection” WWW 2006, May 23–26, 2006, Edinburgh, Scotland. ACM 1-59593-323-9/06/0005.

[16] Tap-dance available online: http://www.atmel.com/dyn/corporate/view_detail.asp?ref=&FileName=Tap-Dance.html&SEC_NAME=Product

[17] Telephone Number Mapping (enum) available online: http://www.ietf.org/html.charters/enum-charter.html

[18] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, Sonesh Surana : “Internet Indirection Infrastructure”, Proceedings of *SIGCOMM* ’02 Pittsburgh, Pennsylvania USA

[19] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan : “Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications”, Proceedings of the 2001 ACM SIGCOMM Conference

[20] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker, “A Scalable Content-Addressable Network” , *SIGCOMM’01,* 2001, San Diego, California, USA

[21] Antony Rowstron1 and Peter Druschel : Pastry: “Scalable, decentralized object location and routing for large-scale peer-to-peer systems”, in Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001). Heidelberg, Germany, November 2001

[22] Parsons, S., Sierra, C., Jennings, N.R.: Agents that reason and negotiate by arguing. Journal of Logic and Computation (1998), 8(3), 261–292.

[23] Foundation for Physical Agents. FIPA Interaction Protocol Library Specification, 2000. Available at http://www.fipa.org

[24] Pitt, J., Guerin, F. and Stergiou, C.: Protocols and Intentional Specifications of Multi-Party Agent Conversations for Brokerage and Auctions. In Proc. Fourth International Conference on Autonomous Agents, ACM Press (2000), 269-276

[25] Reeves, D., Wellman, M. and Grosof, B. Automated Negotiation from Declarative Contract Descriptions. In Proc. Fifth International Conference on Autonomous Agents, (2001)

[26] Esteva, M., Rodriguez, J. A., Sierra, C., Garcia, P., Arcos, J. L.,: On the formal specifications of electronic institutions, In Dignum F. Sierra, C. (eds.) Agent-mediated Electronic commerce (The European AgentLink Perspective), Springer LNAI. (2000)

[27] Wurman, P, Wellman, M. and Walsh W.: A Parameterization of the Auction Design Space, in Games and Economic Behavior, 35 Vol. 1/2 (2001), 271-303

[28] Paschke, A., Dietrich, J., Kuhla, K. A Logic Based SLA Management Framework. Proc of the Semantic Web and Policy Workshop, November, 2005.

[29] Uszok, A., Bradshaw, J.M., Jeffers, R., Johnson, M., Tate, A., Dalton, J., Aitken, S. Policy and Contract Managementn for Semantic Web Services, Proc of the AAAI Spring Symposium on Semantic Web Services, 2004

[30] [RFC 3344] IP Mobility Support for IPv4 available online: http://www.rfc-archive.org/getrfc.php?rfc=3344

[31] M. Buddhikot, G. Chandranmenon, S. Han, Y. W. Lee, S. Miller, L. Salgarelli: “Integration of 802.11 and third-Generation Wireless Data Networks”, Proc. IEEE INFOCOM 2003

[32] Rajiv Chakravorty, Ian Pratt, Pablo Rodriguez : “Exploiting Network Diversity in MARS. A Mobile Access Router System”, available online: http://www.cl.cam.ac.uk/~rc277/hotnets.pdf

[33] Mema Roussopoulos, Petros Maniatis, Edward Swierk, Kevin Lai, Guido Appenzeller, Mary Baker : “Person-level Routing in the Mobile People Architecture”, In Proceedings of the USENIX Symposium on Internet Technologies and Systems, October 1999

[34] Milind Buddhikot, Adiseshu Hari, Kundan Singh, Scott Miller : “MobileNAT : A new technique for mobility across heterogeneous address spaces”, available online: http://www.bell-labs.com/user/mbuddhikot/psdocs/mobilenat-MonetIssue-Buddhikot.pdf

Author Index