UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CLARISSA CASSALES MARQUEZAN

On the Investigation of the Joint Use of Self-* Properties and Peer-To-Peer for Network Management

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Dr. Lisandro Zambenedetti Granville
Advisor

Porto Alegre, November 2010

CIP – CATALOGING-IN-PUBLICATION

Marquezan, Clarissa Cassales

On the Investigation of the Joint Use of Self-* Properties and Peer-To-Peer for Network Management / Clarissa Cassales Marquezan. – Porto Alegre: PPGC da UFRGS, 2010.

228 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2010. Advisor: Lisandro Zambenedetti Granville.

I. Granville, Lisandro Zambenedetti. II. Título.

“Untersuchen was ist, und nicht was behagt.”
“Investigate what is, and not what pleases.”
— JOHANN WOLFGANG VON GOETHE

“Facts are the air of scientists. Without them you can never fly.”
— LINUS PAULING

ACKNOWLEDGMENT

I would like to thank my family for the unconditional support no matter the circumstances, for encouraging me on my professional and personal projects, and for teaching me the great value of studies and perseverance.

To my beloved Kristian Beckers for his affection, patience, and companionship in the last years of my Ph.D.. For understanding the importance of the research in my life and being always supportive even if this means to be far away for a while.

The unconditional support, in good and bad moments, offered by my great friend Viviane Ribeiro Goulart was essential for me. Our numerous conversations always helped me to keep the focus during the period of crises, to give me back the equilibrium and the capacity to sharply see all kind of situations I have been through in the last four years.

I am grateful to the group that we created for having fun and helping each other - the DA (Doutorandos Anônimos - Anonymous Ph.D. students) - whose main members are me, Michelle Denise Leonhardt, and Cristiano Bonato Both, but that was always opened for new desperate Ph.D. students. It was of great importance all our relaxing moments, the information and experiences we exchanged, and the numerous pshilosophical discussions we did about *"what is expected from a Ph.D. student?"*, *"which way to choose?"*, *"how to do the things?"*, and many other issues.

An especial thank is for my great friend Cristiano Bonato Both for being so friendly and comprehensive during all those years. We shared many afflictions related to our Ph.D. experiences, discussions of technical and philosophical aspects of the academic life, and these were moments extremely important for helping me to improve as a professional.

I also want to especially thank my supervisor and great friend Lisandro Zambenedetti Granville for the seven years that we have worked together. All his efforts for encouraging me to improve and go further than the obstacles I faced during my work were more than essential. I thank him for all the lessons, experiences, opportunities, and criticisms. Not always was an easy thing to listen to his advices and criticisms, and, in fact, most of the times I refused to accept them in a first moment. However, today I understand that all this was extremely important to turn me into a researcher.

In addition, I thank the following people from whom I learned a lot: Marcus Brunner, Giorgio Nunzi, Leandro Fortes Rey, Philippe Olivier Alexandre Navaux, Alexandre Carissimi, Rafael Bohrer Ávila, and Tiarajú Asmuz Divério. The experience of working with all of you was very important to me, and the lessons I learned will follow me for life.

To NEC Laboratories Europe, I also thank the great opportunity of doing an internship (sandwich Ph.D.) from April 2008 until March 2009. That was a very intense time of hard working but also learning a lot with people from different parts of the world. The same way, I am grateful for the opportunity of workignan d being [art of the Network and Support Division of the CPD (Data Processing Center) of UFRGS.

Over these four years, many people left and joined the Computer Networks Group. To all these people, I thank the contributions in terms of discussions, friendship, and leisure moments that we spent together. Likewise, I am grateful for the moments I shared with my friends of my former research group GPPD (Grupo de Processamento Paralelo e Distribuído - Parallel and Distributed Processing Group). There are also many other people that helped me in many ways over the last four years. To all these people, my acknowledgment for the intentional and unintentional support.

I also would like to thank the Institute of Informatics of the Federal University of Rio Grande do Sul for the infrastructure provided for the students and for the quality and commitment of their members (*i.e.*, professors, administrative and operational employees).

Finally, I thank the Brazilian agencies devoted to promote the development of research and technology - CAPES and CNPq - for the financial support during my Ph.D. and sandwich Ph.D..

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

4WARD	Forward
AC	Autonomic Computing
ACE	Autonomic Communication Element
AE	Autonomic Element
AM	Autonomic Manager
AMAP	Autonomic Management of Access Points
AMD	Autonomic Management Domain
AME	Autonomic Management Element
AN	Ambient Network
ANA	Autonomic Network Architecture
ANMS	Autonomic Network Management System
AORTA	Autonomic Network Control and Management System
API	Application Programming Interface
AUTOI	Autonomic Internet
B3G	Beyond Third Generation
BIONETS	BIOlogically inspired NETwork and Services
CASCADAS	Component-ware for Autonomic, Situation-aware Communications, And Dynamically Adaptable Services
CBR	Case Based Reasoning
COPS-PR	Common Open Policy Service - Policy Provisioning
COPS	Common Open Policy Service
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unity
DAI	Distributed Artificial Intelligence
DC-SNM	Distributed Cognitive cycle for System & Network Management
DEN-ng	Directory Enabled Networks - next generation

DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Table
DNA	Distributed Network Agent
DNS	Domain Name Service
E3G	Enhanced Third Generation
EFIPSANS	Exposing the Features in IP version Six protocols that can be exploited/extended for the purposes of designing/building Autonomic Networks and Services
FB	Functional Block
FCAPS	Fault, Configuration, Accounting, Performance, Security
FCs	Functional Components
FOCALE	Foundation Observation Comparison Action Learn rEason
GA	Goal Achieved
GENI	Global Environment for Network Innovations
GN	Goal Needed
GPPD	Grupo de Processamento Paralelo e Distribuído
IC	Information Channel
IDP	Information Dispatch Point
IETF	Internet Engineering Task Force
INM	In-Network Management
INMP	In-Network Management Protocol
INMR	In-Network Management Registry
IO	Input Output
IP	Internet Protocol
IPTV	Internet Protocol Television
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IRTF	Internet Research Task Force
ISP	Internet Service Provider
IST	Information Society Technologies
IT	Information Technology
ITU-T	International Telecommunication Union - Telecommunication standardization sector of ITU
JMX	Java Management Extensions
JXTA	Juxtapose

KP	Knowledge Plane
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LLM	Lower Level Manager
M-D-E	Monitoring, Decision-Making, and Execution
MAN	Metropolitan Area Network
MANETS	Mobile Ad hoc Networks
ManP2P	Management Peer-To-Peer
MAPE-K	Monitor, Analyze, Plan, Execute - Knowledge
MAS	Multi Agent Systems
MbD	Management by Delegation
MBTL	Model Based Translation Layer
MCs	Management Capabilities
MIB	Management Information Base
MIMD	Multiple Instruction Stream, Multiple Data Stream
MISD	Multiple Instruction Stream, Single Data Stream
MLM	Middle Level Manager
MONSOON	Multiobjective Optimization for a Network of Sensors using an evOlutionary algOrithm with coNstraints
MP	Management Plane
MPG	Management Peer Group
MRTG	Multi Router Traffic Grapher
NAC	Network Access Control
NDCM	Network Domain Cognitive Manager
NE	Network Element
NECM	Network Element Cognitive Managers
NETCONF	Network Configuration
NMRG	Network Management Research Group
OP	Orchestration Plane
OSKMV	Orchestration, Service, Knowledge, Management, Virtualisation
P2P	Peer-to-Peer
P2PTV	Peer-to-Peer Television
PAN	Personal Area Network
PBNM	Policy Based Network Management

PDC	Parallel and Distributed Computing
PDN	Policy Decision Node
PDP	Policy Decision Point
PEP	Policy Enforcement Point
QoS	Quality of Service
RADIUS	Remote Authentication Dial In User Service
RAS	Reliability, Availability and Survivability
SATOs	Service-aware Adaptive Transport Overlays
Self-CHOP	Self -Configuration -Healing -Optimizing -Protection
Self-FCAPS	Self -Fault, Configuration, Accounting, Performance, Security
Self-NET	Self-Management of Cognitive Future InterNET Elements
SHE	Super Head-End
SIMD	Single Instruction Stream, Multiple Data Stream
SISD	Single Instruction Stream, Single Data Stream
SNMP	Simple Network Management Protocol
SOC	Service-oriented Communication System
SON	Service Overlay Network
SP	Service Enablers Plane
SPMD	Single Program Multiple Data
STB	Set-Top-Box
TCP	Transmission Control Protocol
TEP	Tunnel End Point
TLM	Top Level Manager
UFRGS	Universidade Federal do Rio Grande do Sul
VANET	Vehicular Ad Hoc Network
VHO	Video Hub Office
VoIP	Voice over Internet Protocol
VP	Virtualisation Plane
VSO	Video Serving Office
WAN	Wide Area Network
WBEM	Web-Based Enterprise Management
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
WS	Web Services
WSN	Wireless Sensor Networks

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ABSTRACT

Over the years, the network management community has been pushed towards the design of alternative management approaches able to support heterogeneity, scalability, reliability, and minor human intervention. Currently, there are two major alternatives that have been employed on the design and development of network management solutions. The first one uses autonomic computing and self-* properties, while the second one employs Peer-To-Peer (P2P) concepts and technologies. In general, the investigations related to self-* properties and autonomic computing applied to network management focus their efforts on defining high level models (*e.g.*, ontologies and policies) that are able to determine and drive the autonomous actions of the system. On the other side, P2P research applied to network management is mainly target to define the communication infrastructure of management solution. Thus, in the case of autonomic and self-* properties, there is a lack of investigations approximating the high level models to the management infrastructure, while the P2P investigations suffer from the opposite problem.

Therefore, the investigations carried on this thesis aim at bringing knowledge to issues involving the joint use of self-* properties and P2P to contribute with the development of alternatives for designing network management solutions. The methodology used on the investigations was based on the definition of management requirements, integration issues for the design of the joint use of self-* properties and P2P, and the identification/development of two case studies. Analyzing these case studies, it is was possible to conclude that the first case study (self-healing P2P solution) had an importance in terms of breaking the traditional paradigms of using Management by Delegation on top of P2P infrastructures for network management. Embracing this break through, the second case study (self-organizing P2P solution) gave a step further on the development of P2P application capabilities for the design of self-* properties applied to network management.

The major contributions of this thesis are: (i) the change on the angle of network management solution development from morphological aspects (such as APIs, protocols, architectures, and frameworks) to the design of sophisticated management algorithms; (ii) the introduction of techniques to explore parallel and cooperative behavior of management peers running the management algorithms; (iii) the establishment of design issues that enable the development of truly distributed and cooperative network management environment, where the presence of the human administrator role is minimized and the managers are embedded inside the managed elements and not in the borders of the system. In summary, this thesis shows how to rethink and improve the design and execution of network management tasks.

Keywords: Network management, Self-* properties, P2P, Cooperation.

Investigação do Uso Integrado de Propriedades *Self- e *Peer-To-Peer* para o Gerenciamento de Redes**

RESUMO

Ao longo dos anos, a comunidade de gerenciamento de redes têm sido levada a criar alternativas de gerenciamento que sejam capazes de lidar com problemas de heterogeneidade, escalabilidade, confiabilidade, e com a redução da intervenção humana. Atualmente, existem duas principais alternativas empregadas na definição e desenvolvimento de soluções de gerenciamento de redes. A primeira utiliza computação autonômica e propriedades *self-**, enquanto a segunda utiliza conceitos e tecnologias *peer-to-peer* (P2P). Geralmente, as investigações relacionadas com a aplicação de computação autonômica e propriedades *self-** no gerenciamento de redes são focadas na definição de modelos de alto nível (*ex.*, ontologias e políticas), os quais são capazes de determinar as ações autônomas do sistema. Em contrapartida, pesquisas relacionadas com P2P no gerenciamento de redes estão mais focadas na definição da infraestrutura de comunicação da solução de gerenciamento. Sendo assim, de um lado existem as pesquisas ligadas à aplicação de computação autonômica e propriedades *self-** que sofrem com o problema da falta de aproximação dos modelos de alto nível com a infraestrutura de gerenciamento, e do outro lado existem as que aplicam P2P e que sofrem com o problema oposto.

Dado o cenário descrito acima, essa tese tem como objetivo investigar e esclarecer quais são as questões e características envolvidas na integração de propriedades *self-** e P2P que contribuem para a definição de novas alternativas de soluções de gerenciamento de redes. A metodologia utilizada nas investigações baseia-se na definição de requisitos de gerenciamento, de questões de integração para o uso conjunto de propriedades *self-** e P2P, e na identificação e desenvolvimento de dois estudos de caso. A análise desses estudos de casos mostrou que o primeiro (solução de auto-cura baseada em P2P) é responsável pela quebra do paradigma tradicional do uso de gerenciamento por delegação em cima de infraestruturas P2P. O segundo estudo de caso (auto-organização baseado em P2P) intensifica essa quebra de paradigma ao explorar capacidades de aplicações P2P na definição das propriedades *self-** aplicadas ao gerenciamento de redes.

As maiores contribuições dessa tese são: (i) a mudança no foco no desenvolvimento das soluções de gerenciamento de redes dos aspectos morfológicos (tais como, APIs, protocolos, arquiteturas, e *frameworks*) para a definição de algoritmos sofisticados de gerenciamento; (ii) a introdução de técnicas para explorar comportamentos paralelos e cooperativos dos *peers* de gerenciamento que executam tais algoritmos; (iii) a definição dos *design issues* que possibilitam o desenvolvimento de ambientes de gerenciamento de redes verdadeiramente distribuídos e cooperativos, onde o papel dos administradores humanos é minimizado e os gerentes estão embutidos dentro dos elementos gerenciados e não na borda dos sistemas. Em resumo, essa tese mostra como repensar a definição e execução de tarefas de gerenciamento de redes.

Palavras-chave: Gerenciamento de redes, Propriedades *self-**, P2P, Cooperação.

1 INTRODUCTION

Network management is an important discipline whose main goal is to maintain the communication infrastructures and network services working in a proper manner. Along the years, several challenges have been faced by the research community on network management. Examples of such challenges are the definition of standard network management protocols, *e.g.*, Simple Network Management Protocol (SNMP) (HARRINGTON; PRESUHN; WIJNEN, 2002), Common Open Policy Service (COPS) (PEREIRA; GRANVILLE, 2008), NETCONF (Network Configuration) (BHUSHAN; SCHÖNWÄLDER, 2009); the investigation of approaches to proceed with network management operations, *e.g.*, centralized and distributed (MARTIN-FLATIN; ZNATY; HABAUX, 1999) (SCHÖNWÄLDER; QUITTEK; KAPPLER, 2000) (PAVLOU, 2007). Although the research proposed so far have introduced improvements on network management, there are several new challenges (*e.g.*, reduction of human intervention, heterogeneity, scalability, reliability) that are not completely covered by the previous management approaches. The research community, in this way, started to investigate new alternatives for designing and developing network management solutions (SCHÖNWÄLDER et al., 2006).

One of the alternatives investigated is the employment of autonomic computing in network management (SAMAAN; KARMOUCH, 2009). The term Autonomic Computing (AC) has been first used in 2001 (HORN, 2001) and its most accepted definition was presented by Kephart and Chess (KEPHART; CHESS, 2003). The key issues behind autonomic computing are the reduction of human intervention and increasing the autonomous behavior of the systems (*e.g.*, self-configuration, self-monitoring). Considering the network management community, it is also possible to evidence the use of other terms like autonomic communications, self-management, and self-* properties, to refer to the key issues related to autonomic computing (SAMAAN; KARMOUCH, 2009) (STRASSNER et al., 2009) (BOUABENE et al., 2010) (MANZALINI; ZAMBONELLI, 2006). Another alternative explored by the research community is the employment of Peer-to-Peer (P2P) technologies on network management solutions. The use of P2P has been claimed to be a powerful alternative for enhancing connectivity across heterogeneous domains, scalability, reliability, and cooperation among managers (GRANVILLE et al., 2005) (XU et al., 2008) (FIORESE; SIMÕES; BOAVIDA, 2009). In fact, the literature shows that the investigations on self-* properties and P2P technologies constitute two major alternatives, currently, followed by the research community on network management.

In general, the investigations related to self-* properties and autonomic computing applied to network management focus their efforts on defining high level models (STRASSNER et al., 2008) (BERGLUND et al., 2008)(CHONG et al., 2009), such as ontologies (SERRANO; SERRAT; STRASSNER, 2007) (KEENEY; LEWIS; SULLIVAN, 2007) (FUENTES; VERGARA; CASTELLS, 2006) and policies (SIMMONS; LUTFIYYA,

2005) (MEER et al., 2006) (ZHAO; CHEN; CRESPI, 2008) (HADJIANTONIS; PAVLOU, 2008), that are able to determine and drive the autonomous actions of the system. In contrast, P2P research applied to network management is mainly targeted to define the communication infrastructure of management solution (GRANVILLE et al., 2005) (BARSHAN; FATHY; YOUSEFI, 2009), infrastructure to support information dissemination (YALAGANDULA et al., 2006) (BINZENHÖFER et al., 2006), and the connectivity of the management elements (*e.g.*, managers, agents, devices) (ZHOU; RENESSE, 2005) (FALLON et al., 2007). Thus, in the case of self-* properties, there is a lack of investigations approximating the high level models to the management infrastructure, while the P2P investigations suffer from the opposite problem. In this sense, self-* properties and P2P constitute complementary techniques, and the joint use of those techniques could be exploited as another alternative for the development of network management solutions.

Currently, there exists research showing the feasibility of the joint use of self-* properties and P2P technologies. For instance, there are some proposals of using self-* properties to provide a better management of the P2P network (BEJAN; GHOSH, 2004) (SACHA et al., 2006) (JONES et al., 2009). Other proposals, for example, employ self-* properties to enhance the performance of the P2P systems (NTARMOS; TRIANTAFILLOU, 2005) (BISKUPSKI; DOWLING; SACHA, 2007) (XIE; MIN; DAI, 2009). However, the employment of self-* properties and P2P in order to build network management solutions has not been, so far, extensively investigated, and the potentialities and drawbacks of this union remains unclear. Therefore, this thesis aims at bringing knowledge to issues involving the joint use of self-* properties and P2P to contribute with the development of an alternative for designing network management solutions. To drive the processes of bringing knowledge, this thesis establishes a line of investigation based on hypothesis and fundamental questions, which are described as follows.

1.1 Hypothesis and Fundamental Questions

Over the years, the development of network management solutions was focused on providing protocols, architectures, frameworks, and APIs (Application Programming Interface) - *i.e.* morphological aspects - while the design of the management application itself was always neglected. Pushed by the posed challenges, the network management community started to employ more sophisticated solutions, like the two ones described above: self-* properties and P2P techniques. However, most of those sophisticated solutions were still focused on the morphological aspects of network management. In parallel, discussions about the lack of truly distributed and cooperative solutions, and the lack of investments on algorithms started to be emphasized by the network community (SCHÖNWÄLDER et al., 2006), but no clear solution was proposed so far. Therefore, to overcome the focus on morphological aspects and find the key to truly distributed and cooperative solution, this thesis present the following hypothesis.

***Hypothesis.* "The combination of self-* properties and P2P techniques enables the design and development of truly distributed and cooperative network management applications."**

In order to guide the investigation of this thesis, fundamental questions associated with the hypothesis are defined and presented as follows.

Fundamental question I. *What are the characteristics introduced by the self-* properties and P2P techniques on the design and execution of network management solutions?*

Fundamental question II. *What are the benefits of employing self-* P2P techniques on building network management solutions in comparison to self-* centralized or self-* hierarchical approaches?*

Fundamental question III. *Which kind of costs does the self-* P2P-based network management approach impose?*

The methodology to be employed to carry out the investigation is based on the development of case studies and accomplishment of integration issues on the design of self-* properties and P2P techniques for the network management solutions. Two case studies are defined. The first one investigates the employment of the self-healing property and P2P for the fault management of management platforms. The other explores the joint use of self-organizing property and P2P interactions in order to manage the amount of network traffic in network virtualization environments. Each one of the case studies has been evaluated following different research methods (*e.g.*, implementation and simulation). Then, based on the observation of the characteristics of the solutions developed for each case study and its associated results, it is possible to derive outcomes related to the joint use of the self-* properties and P2P-based approach on network management. These outcomes are analyzed in order to answer the fundamental questions of the thesis. In addition, the answers of such questions also constitute the contributions of this thesis for the network management research area.

The organization of this thesis is described in the next section.

1.2 Organization

Chapter 2 describes the state of the art on the three topics that are addressed by this thesis: network management, autonomic computing and self-* properties, and peer-to-peer. In addition, the relationship among these topics is also analyzed, resulting in the description of: network management and self-* properties, network management and P2P, and, finally, the combination of those three topics.

Chapter 3 depicts the principals of this thesis. Five topics are addressed in this chapter. First, the conditions that lead to the need of a self-* P2P alternative are presented. Second, it is presented a description of the network environments that might benefit from self-* P2P solutions and four management requirements are proposed to be associated to the need of a self-* P2P alternative. Third, terms and concepts used in the context of the self-* P2P alternative are defined. Fourth, integration issues are proposed in this thesis to be used to guide the design of self-* P2P solutions. Finally, case studies are identified to be tested under the light of self-* P2P alternative.

Chapter 4 describes the case study associated with the joint use of self-healing and P2P techniques. The solution designed to integrate those techniques is presented, as well as the scenario used to validate the execution of this solution. In this case study, the scenario is a Network Access Control (NAC) monitoring system.

Chapter 5 presents the investigation of self-organizing and P2P in order to manage the network traffic of network virtualization environments. A self-organizing P2P approach is designed and instantiated. The evaluation of such an approach considers virtual IPTV providers on top of the substrate network.

Chapter 6 shows the results discussion related to the ideas that can be derived from the investigations performed in this thesis. Those discussions address the relationship of the self-* P2P approach with parallel and distributed computing area, the analysis of the integration issues and its hidden characteristics, and, finally, presents the placement of the self-* P2P approach under research scenario of network management discipline.

Chapter 7 presents the final remarks and conclusions associated to this thesis. The answers for the fundamental questions and contributions are exposed and justified. In addition, opportunities to develop future works are identified and detailed.

2 STATE OF THE ART

This thesis touches three different areas: network management, autonomic or self-* properties, and peer-to-peer. Indeed, the key of the work developed here is the integration of those three areas. Therefore, understanding the start of the art associated with each area is of crucial importance to comprehend how their integration can be accomplished. In a first moment, this chapter presents the current research status of each area. In a second moment, current investigations are examined in order to show the relationship among those areas, *i.e.*, autonomic computing/self-* properties and network management, peer-to-peer and network management, and, finally, all three areas together.

2.1 Background

As emphasized in Chapter 1, the investigation of new approaches and alternatives for developing network management solutions has gained attention over the years. For this reason, Section 2.1.1 shows the approaches and alternatives developed so far. Instead of listing isolated works, this section shows, initially, the types of taxonomies proposed in the literature, that try to classify the proposed network management approaches. Then, the most popular approaches found in the literature are described. Finally, it is presented the discussions of the network management community about the problems left unsolved by those approaches and the expectations for further alternatives.

In the sequence, basic concepts related to autonomic computing and self-* properties are presented. Autonomic computing and self-* properties comprise an area of research that is used in different fields of computer science, (*e.g.* business (GREENWOOD, 2008), data bases (ZEWDU; DENKO; LIBSIE, 2009), and cloud computing (MURPHY et al., 2010) (KIM et al., 2009)). In addition, there are no standard definitions of terms and models. Therefore, discussions, terms, and concepts related to autonomic/self-* properties applied to network management field are described in Section 2.1.2.

Finally, the description of current research related to peer-to-peer is presented in Section 2.1.3. Peer-to-peer is employed in different contexts, is associated with different levels of abstraction, and interpreted in distinct manners (ANDROUTSELLIS-THEOTOKIS; SPINELLIS, 2004). Thus, the objective of this section is to clarify the contexts and concepts where P2P appears.

2.1.1 Network Management Approaches

On network management community, there are no standardized classification or taxonomy of network management approaches. Over the years, some attempts to organize those approaches were proposed, and are presented below.

Martin-Flatin *et.al.* (MARTIN-FLATIN; ZNATY; HABAUX, 1999) proposed a simple and an enhanced taxonomies. The former is based on the organization criterion and divides the network management approaches in: centralized; weakly distributed hierarchical; strongly distributed hierarchical; and strongly distributed cooperative. The later is based on four criteria (delegation granularity, semantic richness of the information model; degree of automation of management; degree of specification of a task) and presents seven categories: no delegation; delegation by domain; delegation by micro-task with low-level semantics; delegation by micro-task with high-level semantics; delegation by macro-task with low-level semantics; delegation by macro-task with high-level semantics; delegation by macro-task with very high-level semantics.

Schönwälder *et.al.* (SCHÖNWÄLDER; QUITTEK; KAPPLER, 2000) introduced a taxonomy very similar to the previous one. However, the authors categorized the network management approaches considering solely the number of managers used on the system. This taxonomy has four management categories: centralized; weakly distributed; strongly distributed; and cooperative.

George Pavlou (PAVLOU, 2007) defined a taxonomy based on the different ways of executing a management task. The proposed taxonomy organizes management approaches, frameworks, and protocols in three levels. The first one divides the management approaches in: remote invocation and management by delegation. On the second level, remote invocation is divided into manager-agent based and object/service interface based; and the management by delegation is formed by manager-agent based and full mobile code based. The third level describes the protocols associated with each branch of the taxonomy.

The aforementioned taxonomies have in common the presence of different types and levels of distribution. In fact, with the challenges that were emerging over the years (*e.g.*, number of managed elements, heterogeneity, reliability), and the evolution of the network management area, it has become a common sense on the network management community the fact that, in general, distributed approaches are more suitable than centralized ones. For instance, during the meeting of the Network Management Research Group (NMRG) of the Internet Research Task Force (IRTF), in October 2006 (SCHÖNWÄLDER et al., 2006), one of the main issues listed by researchers, vendors, and operators was the definition of new distributed approaches for network and service management. Below, the common distributed network management approaches are presented.

Mobile Agents. This approach enables the migration of the management code from the managers to the managed network elements. Mobile agents are able to execute an on-demand and customized distribution of the configuration and management programs (DU; LI; CHANG, 2003) (STEPHAN; RAY; PARAMESH, 2004) (KOCH et al., 2004) (TO; KRISHNASWAMY; SRINIVASAN, 2005) (GUO; ZENG; CUI, 2009) (ZHENG; DONG, 2009).

Management by Delegation (MbD). Goldszmidt and Yemini (GOLDSZMIDT; YEMINI, 1995) proposed this distributed network management approach. The key concept behind MbD is the employment of delegated-agents that are responsible for executing management tasks on the devices instead of bringing data from the devices to the network management platform-based applications. In this sense, delegation can be used to move management functions to the data rather than move

data to these functions. Originally the management functions were customizable scripts, that were written in “ah doc”manners. Schönwälder *et al.* (SCHÖNWÄLDER; QUITTEK; KAPPLER, 2000) introduced the employment of Management Information Base (MIB) (MCCLOGHRIE; ROSE, 1991) for the definition of such scripts, so called ScriptMIB. Based on MbD approach several investigations were developed (ROCHA; ROCHA; SOUZA, 2004) (FIOREZE et al., 2005) (STRAUSS; SCHÖNWÄLDER; QUITTEK, 2001) (CHERKAOUI et al., 1998).

Policy Based Network Management (PBNM). The goal of policy-based management is to govern the behavior of a system based on the definition of high level policies (SLOMN, 1994). One of the most accepted architecture for PBNM was defined by the Internet Engineering Task Force (IETF) (WESTERINEN et al., 2001). This architecture is composed of four main components: policy tool, policy repository, Policy Decision Point (PDP), and Policy Enforcement Point (PEP). The policy tool is the administrator front-end from where the management policies are defined and edited, to be then stored in the policy repository for future use. When deploying a policy, the policy tool signs the PDPs to retrieve the policy from the repository and translate it to configuration commands on the PEPs (*e.g.*, network interfaces, queuing disciplines) located inside the network devices. This approach has been employed in multiple network management scenarios (WRIGHT, 1999) (ZHAO; CHEN; CRESPI, 2008) (FIOREZE et al., 2006) (ALVES et al., 2006).

Multi-agents. According to Timm *et al.* (TIMM et al., 2006) multi-agent systems are composed of heterogeneous agents that are generally self-motivated and act to fulfill internal goals, but may also share tasks with others. There is no global or centralized control mechanism and agents have to reason to co-ordinate their actions, plans, and knowledge. Such agents are also referred as intelligent agents (TRZEC; HULJENIC, 2002). In general, the employment of multi-agents enables the decomposition of network management tasks into sub-tasks that are executed by the agents spread along the network (LIU; LI; LUO, 2004) (M. et al., 2008) (LEE et al., 2004) (ZHANG et al., 2008) (LUO; LI; LIU, 2006).

Peer-To-Peer. Since 2003, the use of P2P in network management solutions, also called as P2P-based Network Management, has been explored by several proposals (STATE; FESTOR, 2003) (GONG, 2005) (GRANVILLE et al., 2005) (BARSHAN; FATHY; YOUSEFI, 2009) (BINZENHÖFER et al., 2006) (XU et al., 2008) (KAMIENSKI et al., 2006) (BRUNNER et al., 2005) (LENG et al., 2007). The major advantages of using this approach are the scalability, availability, reliability, and connectivity of the network management solutions. The details about the employment of P2P in network management are presented in Section 2.3.

In addition to the approaches listed above, there are technologies that contributed for the evolution and dissemination of distributed network management. Some of these contributers are: Web technologies (BARILLAUD; DERI; FERIDUN, 1997), Common Object Request Broker Architecture (CORBA) (OBJECT MANAGEMENT GROUP, 2010), and Web Services (WS) (CURBERA et al., 2003). Based on these technologies other distributed network solutions were developed, like for example: Web-based network management platforms (ANEROUSIS, 1999) (MULLER, 1997); CORBA was used to enable distribution and remote invocation of management tasks (SCHULZE et al., June) (CUI;

GUTIÉRREZ, 2003); and WS enabled interoperability among heterogeneous and distributed network solutions, composition, and is also referred as WS-based network management (ROHR; GRANVILLE; TAROUCO, 2009) (VIANNA et al., 2007) (MOURA et al., 2007) (SOLDATOS; ALEXOPOULOS, 2007) (ZHANG et al., 2006). In fact, Web, CORBA, and WS have become well established technologies employed in network management solutions. In contrast to this, and as expected, new emerging technologies and concepts (*e.g.*, overlays (TANG; AL-SHAER, 2008) (CAPONE; ELIAS; MARTIGNON, 2008), Mashups (XU; SONG; LUO, 2009) (BEZERRA et al., 2009)(BEZERRA et al., 2010), and virtualization (WANG et al., 2008)) have been investigated in order to enhance the development of network management solutions.

Despite the diversity of approaches and technologies employed so far in network management, what remains is the feeling that the solutions developed so far are not able to properly fulfill the current needs on network and services management. For example, mobile agents failed, according to Rolf Stadler (STADLER, 2006), because at the time they were investigated the basis of Distributed Artificial Intelligence (DAI) area could not deliver important issues for the network management field (*e.g*, interdomain negotiations and failure detection). In contrast, MbD and PBNM became very popular approaches, but they suffer from scalability and reliability limitations. To overcome the problems of the earlier approaches, multi-agents[1] and P2P started to be investigated. They do have the potential to provide fully distributed solutions able to tackle the challenges of network and service management area. However, they were employed so far to build up management platforms and not management applications. In summary, the members of the NMRG of IRTF (SCHÖNWÄLDER et al., 2006) highlighted that the desirable topics to be investigated on the design of future network and services management approaches are: fully distributed solutions (*e.g.* P2P); change the focus from data structures and protocols towards algorithms; increase the efforts on developing cooperative management solutions; and the investment of research on self-* technologies for network management.

2.1.2 Autonomic Computing and Self-* Properties

Since the first time that the term autonomic computing was used until now, many points of view and definitions were formulated (KEPHART; CHESS, 2003) (STRASSNER et al., 2009) (BOUABENE et al., 2010). In fact, there are no standardized definitions concerning the terms autonomic communications, self-management, and self-* properties. For example, Kephart's and Chess's vision of autonomic computing (KEPHART; CHESS, 2003) omits the notion of communication. In contrast, Bouabene *et al.* (BOUABENE et al., 2010) proposes an autonomic computing approach that explicitly considers communications.

The broadness of the term autonomic computing is also a topic of divergence. According to Dobson *et al.* (DOBSON et al., 2010), IEEE and other organizations employ this term to describe *"the application of advanced technology on the management of advanced technology"*. Dobson *et al.* also listed examples of visions related to autonomic computing, such as organic computing (GUDEMANN et al., 2008), bio-inspired computing (CHIANG; BRAUN; AGBINYA, 2007), self-organizing systems (AL-OQILY; KARMOUCH, 2008), autonomous and adaptive systems (LEIBNITZ; WAKAMIYA; MURATA, 2006). Indeed, the term autonomic computing on Dobson's *et al.* perspective encompasses all the aforementioned visions (DOBSON et al., 2010).

[1]Research on multi-agents is in fact the evolution of research on DAI (WEISS, 1999).

In the literature, there are some initiatives trying to organize the characteristics, concepts, and visions associated to autonomic computing. For example, Hariri *et al.* (HARIRI et al., 2006) and Lin *et al.* (LIN; MACARTHUR; LEANEY, 2005) described the characteristics of autonomic computing (both based on the original concept of Paul Horn (HORN, 2001)), which are listed below.

1. **Self-Awareness.** An autonomic system knows itself and is aware of its behavior.
2. **Self-Protecting.** An autonomic system is prone to attacks and hence it should be capable of detecting and protecting its resources from both internal and external attack and maintaining overall system security and integrity.
3. **Self-Optimizing.** An autonomic system should be able to detect performance degradation in system behavior and intelligently perform self-optimization functions.
4. **Self-Healing.** An autonomic system must be aware of potential problem and should have the ability to reconfigure itself to continue to function properly.
5. **Self-Configuring.** An autonomic system must have the ability to dynamically adjust its resources based on its state and the state of its execution environment.
6. **Context Aware.** An autonomic system must be aware of its execution environment and be able to react to changes in the environment.
7. **Open.** An autonomic system must be portable across multiple hardware and software architectures, and must be built on standard and open protocols and interfaces.
8. **Anticipatory.** An autonomic system must be able to anticipate its needs and behaviors and be able to manage itself pro-actively.

Taking the aforementioned characteristics as a comparison basis, Lin *et al.* (LIN; MACARTHUR; LEANEY, 2005) presented a list of different autonomic computing definitions, and which kind of characteristics each one presents. Table 2.1 reproduces the table illustrated by Lin *et al.* with the addition of recent works on the area. The numbers of Table 2.1 represent the aforementioned autonomic computing characteristics.

Table 2.1: List of different definitions for autonomic computing and their characteristics

	Characteristics							
First Author	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Horn (HORN, 2001)	✓	✓	✓	✓	✓	✓	✓	✓
Kephart (KEPHART; CHESS, 2003)		✓	✓	✓	✓		✓	✓
Sterritt (STERRITT; BUSTARD, 2003)	✓	✓	✓	✓	✓	✓	✓	✓
Ganek (GANEK; CORBI, 2003)	✓	✓	✓	✓	✓	✓	✓	✓
Kaiser (KAISER et al., 2003)		✓	✓	✓				
Agrawal (AGARWAL et al., 2003)	✓	✓	✓	✓	✓	✓	✓	
Trumler (TRUMLER et al., 2004)		✓	✓	✓	✓	✓		✓
Dobson (DOBSON et al., 2010)	✓	✓	✓	✓	✓	✓		

The diversity on autonomic computing is not restricted to its definition. Likewise, a different number of theories and technologies have been employed to develop autonomic

solutions. Khalid *et al.* (KHALID et al., 2009) proposed to aggregate the research work focusing on autonomic frameworks, architectures, and infrastructures into seven groups, that are described below.

Biologically inspired frameworks and architectures. The objective is to mimic various biological systems (*e.g.* ant-colonies, food chains, interactions among populations, among others). According to Khalid *et al.* (KHALID et al., 2009), both centralized and distributed approaches have been suggested. The centralized ones focus on the role of human nervous system as a controller to regulate and maintain the other systems (HARIRI et al., 2006) (STERRITTA et al., 2005). On the other hand, decentralized approaches use inspiration from cell and ant-colonies (ANALOUI; JAMALI, 2008) (HONG et al., 2008) (CHAKRAVARTI; BAUMGARTNER; LAURIA, 2005).

Large scale distributed applications frameworks. Solutions on this group are focused on providing self-configurable and self-organizing infrastructures able to deliver high availability and scalability. Examples of works in this group are: Oceano (APPLEBY et al., 2001) and OceanStore (RHEA et al., 2001), IBM's SMART (LOHMAN; LIGHTSTONE, 2002) and Microsoft's AutoAdmin (CHAUDHURI; NARASAYYA, 2007).

Frameworks using agent architecture. Frequently used to develop infrastructures to support autonomic behavior. Agent architecture uses a decentralized approach, where each agent has its own local control and interacts with the other agents to create a self-managed system. (DONG et al., 2003) (WOLF; HOLVOET, 2003) (RAO; GHENNIWA; SHAMI, 2007). The notion of cooperation of individual elements to achieve a common goal is a fundamental aspect of multiagent system research, and therefore multiagent are commonly used on autonomic computing research (TESAURO et al., 2004). However, Huebscher and McCann (HUEBSCHER; MCCANN, 2008) remarked that implementing cooperative autonomic elements with multiagent systems suffers from the difficulties in guaranteeing that the behavior emerging from individual goals of each agent will truly represent the common goal of the autonomic system. One alternative typically employed to avoid dealing with the multiagent cooperation is a hierarchical structure of the autonomic elements.

Technique focused frameworks. There are several frameworks that base their solution on techniques from Artificial Intelligence such as BDI logic model[2] (PENG et al., 2009) and predicate model (RANGANATHAN; CAMPBELL, 2004); control theory (DIAO et al., 2005); ontologies (KEENEY; LEWIS; SULLIVAN, 2007); policies (BALIOSIAN et al., 2008); ontologies and policies (STRASSNER et al., 2009); genetic algorithms (GELENBE; LIU; LAINE, 2006).

Component based frameworks. The key behind the component based frameworks is to create components that enable an autonomic behavior (VIROLI; CASADEI; OMICINI, 2009) (WANG et al., 2008) (MALATRAS; PAVLOU, 2007) (BAUDE; HENRIO; NAOUMENKO, 2007).

Self-managed service oriented architecture. In general, proposals use service oriented architecture to enable autonomic behavior (GURGUIS; ZEID, 2005) (LIU et al., 2005) (ZHANG, 2007) (CALLAWAY et al., 2008).

[2]BDI - Belief-Desire-Intention.

Infrastructure for injecting autonomicity into non-autonomous systems. There are proposals that create an infrastructure to inject autonomic behavior in legacy non-autonomic systems. Some of the techniques used by those approaches are layered architectures (ANTHONY; BUTLER; IBRAHIM, 2005), Case Based Reasoning (CBR) (KHAN; AWAIS; SHAMAIL, 2008), and rule driven approach (STANFEL; HOCENSKI; MARTINOVIC, 2007).

Khalid *et al.* (KHALID et al., 2009) also identified two main design approaches that are followed by the autonomic, self-management proposals. The first one is the **Externalization Approach** where modules enabling self-management lie outside the managed system. The other one is the **Internalization Approach** where the self-management of the application is done inside the managed system. In the authors perspective, externalization approach is more effective because it uses separated modules to let the problem detection and resolution localized in such modules.

In addition to the general discussions about autonomic computing and self-management, there are specific communities developing their own ideas and definitions. For example, after several years of discussion, the network management community seems to be converging into a common understanding of what represent the terms autonomic computing systems and Autonomic Network Management Systems (ANMS). There are two works (one from Samaan and Karmouch (SAMAAN; KARMOUCH, 2009), and the other from Huebscher and McCann (HUEBSCHER; MCCANN, 2008)) defining autonomic computing, system, or behavior as a self-managed system presenting the self-CHOP properties, *i.e.*, self-configuring, -healing, - optimizing, and -protection. Based on this autonomic computing description, Samaan and Karmouch (SAMAAN; KARMOUCH, 2009) also defined the ANMS term as a network management system that employs the autonomic computing concept. Thus, an ANMS must perform management operations following the self-CHOP properties.

In fact, the network management community is depositing efforts on trying to define models, architectures and standards for developing autonomic communication systems. Examples of such efforts are proposals like Focale (STRASSNER et al., 2009), Autonomic Network Architecture (BOUABENE et al., 2010), CASCADAS (MANZALINI; ZAMBONELLI, 2006), among others. However, none of those proposals has a large acceptance and is recognized as a '*de facto*' standard. So, due to the lack of well-established terminologies, models, and architectures, it is possible to observe a proliferation of solutions for specific types of networks and purposes (BALASUBRAMANIAM et al., 2006) (FALLON et al., 2007) (ZHOU; LYU, 2007). The investigations of autonomic or self-* solutions applied to network management are detailed in Section 2.2.

2.1.3 Peer-To-Peer

The term Peer-to-Peer (P2P) can be applied to several and distinct contexts. In fact, the analysis of the literature shows that the term P2P can be accompanied with words like system, application, infrastructure, overlay, and networks. According to Androutsellis-Theotokis and Spinellis *"it is fair to say that there is not a general agreement on what 'is' and what 'is not' peer-to-peer"* (ANDROUTSELLIS-THEOTOKIS; SPINELLIS, 2004). These authors attribute the lack of agreement to the fact that systems or applications are labeled peer-to-peer not because their internal behavior, but due to their external appearance. Despite the fuzzy definitions and terminologies associated with P2P technology, Androutsellis-Theotokis and Spinellis grouped these technologies into: P2P infrastructures and P2P applications (ANDROUTSELLIS-THEOTOKIS; SPINELLIS, 2004).

They defined these groups considering only one kind of application, the P2P content distribution, which was the most popular and developed technology at the time their paper was published. However, over the years, other types of P2P technologies emerged. Investigating the literature it is possible to analyze and regroup such technologies into: P2P infrastructures, P2P application, and P2P infrastructures for specific applications[3]. Examples related to each one of these groups are presented below.

P2P infrastructures. Technologies developed to build underlying conditions and services to support applications. Examples of such conditions and services are: routing and location (KAUNE et al., 2008) (CHENG; YUKSEL; KALYANARAMAN, 2009) (KANG et al., 2009), reputation (SANCHEZ-ARTIGAS; GARCIA-LOPEZ, 2009), topology management (PAPADAKIS et al., 2009) (AGHAZADEHY et al., 2009), performance (ZINNER et al., 2008) (XIE; MIN; DAI, 2009), connectivity (WACKER et al., 2008) (JIMENEZ; OSMANI; KNUTSSON, 2009), security (LUA J. CROWCROFT et al., 2008) (JYOTHI; DHARANIPRAGADA, 2009). Some of the well-known P2P infrastructures are, for instance, JXTA (GONG, 2001) (XHAFA et al., 2008), Pastry (BJUREFORS; LARZON; GOLD, 2004), and Chord (STOICA et al., 2003).

P2P applications. In general, the proposals of this group are applications that make use of P2P infrastructures. Examples of current popular P2P applications are: file sharing (FAN; LUI; CHIU, 2009) (DHURANDHER et al., 2009) (MEULPOLDER et al., 2009); multimedia streaming (BARBERA et al., 2007) (CHEN et al., 2009) (LIU; RILEY, 2009); P2P Television (P2PTV) (LIU; LI, 2008) (ALHAISONI; LIOTTA, 2009); and searching documents and databases (LIN et al., 2007) (LI; SHOU; TAN, 2008) (DONG; YUE-LONG, 2009);

P2P infrastructures for specific applications. This group of technologies is comprised of investigations that present a very tight relationship between the P2P infrastructure and the application running on top of such infrastructure. Based on the analysis of the literature, it is possible to list some proposals belonging to this group, such as multiplayer games (SCHMIEG et al., 2008) (VARVELLO; DIOUT; BIERSACK, 2009); workflow (GAO; ZENG, 2009); Voice over IP (VoIP) (CHEN; WANG; JAJODIA, 2006)(SANGHAN; HASAN, 2007) (TIRASOONTORN; KAMOLPHIWONG; SAE-WONG, 2008) (ZHANG et al., 2009); and typical P2P applications such as file sharing (DHIWAL et al., 2008) (ALTMANN; BEDANE, 2009) and multimedia (MATHIEU; PARIS, 2009).

Androutsellis-Theotokis and Spinellis (ANDROUTSELLIS-THEOTOKIS; SPINELLIS, 2004) also classified P2P applications into five categories based on the purposes associated with the applications. The analysis of recent literature reveals that the definition of those categories is still valid. The five categories are described below.

Communication and collaboration. These applications usually focus on providing direct communication among peers (*e.g.*, instant messaging applications like Google Talk (INC., 2009a)). The possibility of direct communication can enable collaborative behaviors.

[3]It is out of the scope of this thesis to define a new categorization of P2P technologies.

Distributed computation. In this category, it is possible to find applications that need to compute massive tasks. For doing this, such applications break-down the tasks into small ones and distribute them among the available peers of the P2P infrastructure (*e.g.*, Seti@Home (KORPELA et al., 2001), Einstein@Home (EINSTEIN@HOME, 2009), Rosetta@Home (LABORATORY; WASHINGTON; COMMONS, 2009), Quake-Catcher Network (UNIVERSITY; RIVERSIDE, 2009)).

Internet service support. This group is composed of applications that use P2P infrastructures to provide services like: video conference (*e.g.*, Qnext (CORPORATION, 2009)), telecommunication (*e.g.*, Skype (LIMITED, 2009)), Web portals (*e.g.*, Osiris (TEAM', 2009)), streaming (*e.g.*, PPLive (INC., 2009b), PeerCast (PEERCAST.ORG, 2009), TVU Network (SHEN, 2009)), among others.

Database systems. The applications of this group are able to use the P2P infrastructure as a database system, instead of a traditional central repository (BANAEI-KASHANI; SHAHABIA, 2008) (DONG; YUE-LONG, 2009).

Content distribution. This is the most popular category of applications. In this case, files are spread along the P2P infrastructure and can be accessed through file sharing or content distribution application (*e.g.*, Mininova (MININOVA, 2009), eMule (TEAM, 2009), KaZaA (NETWORKS, 2009)).

Some of the major contributions of P2P for the society and for the research community are related to the variety of applications that can be developed exploring (i) the features introduced by P2P infrastructure (*e.g.*, scalability, robustness, and reliability); and (ii) the design concepts behind the P2P applications (*e.g.*, distributed algorithms, collaboration on executing a task, sharing information, decentralization of decisions). Encouraged by the features and design concepts introduced by P2P approaches, the network management community started to explore those approaches on their solutions. The discussion about the use of P2P on network management solutions is described in Section 2.3.

2.2 Autonomic Computing and Network Management

The research developed on autonomic computing and self-* properties (or self-star) applied to network management can be divided in two major groups. The first one comprises high level architectures for deploying autonomic networks, autonomic communications, and/or self-* based network management. The second group is related to specific networks, services, and management tasks.

2.2.1 Architectural Approaches

The investigation of architectural approaches for autonomic network management can be divided into two groups. The first one is related to proposals in the context of projects and the second one comprises individual attempts. Some of the main projects devoted to build autonomic network management architectures are described below, and in the sequence some independent initiatives are presented.

Foundation Observation Comparison Action Learn rEason (FOCALE) is an autonomic network management approach that is meant to be built on top of the current established network management environments (STRASSNER; AGOULMINE; LEHTIHET, 2006). The basis of FOCALE are ontologies, policies, and

context-aware mechanisms to provide self-knowledge for the autonomic system (JENNINGS et al., 2007) (STRASSNER et al., 2009). The architecture is based on the concept that business objectives, user requirements, and environment context change dynamically. Thus, to handle such kind of changes, FOCALE employs distribution and two control loops: a maintenance control loop and an adjustment control loop. Jennings et al. (JENNINGS et al., 2007) describe the elements composing the architecture: Autonomic Management Element (AME), Autonomic Manager (AM), Model Based Translation Layer (MBTL), Autonomic Management Domain (AMD). The coordination of management decision making is performed by the AMEs. The information model employed is based on Directory Enabled Networks - next generation (DEN-ng) (STRASSNER, 2002). Policy based network management is also incorporated in FOCALE architecture.

Autonomic Network Architecture (ANA) is a project whose main objective is to enable networks to scale in size and functionalities (BOUABENE et al., 2010). The core networking elements of ANA architecture are: Information Dispatch Point (IDP), Functional Blocks (FB), and Information Channels (IC). The concept of compartments was defined in order to enable communication between hosts and routers implementing and using the same set of functional blocks. A compartment is a set of FBs, IDPs and ICs with some commonly agreed set of communication principles, protocols and policies (ANA, 2009). Jelger et al. (JELGER et al., 2007) presented the principles of ANA architecture, and different from traditional approaches, ANA considers that heterogeneity is a basic element of the network. This architecture provides manners to make the network adapts itself to deal with heterogeneous styles and demands. In addition, Zseby et al. (ZSEBY et al., 2009) argue that the basic mechanism to establish autonomic communication principles (present in ANA project) is to find solutions for the situation awareness problem. The solution of such problem involves perception of conditions and events, collaboration, coordination, and several other issues pointed by the authors.

Component-ware for Autonomic, Situation-aware Communications, And Dynamically Adaptable Services (CASCADAS) is an IST project aiming at developing and validating an autonomic framework for creating, executing, and provisioning situation-aware and dynamically adaptable communication services (MANZALINI; ZAMBONELLI, 2006). According to Manzalini *et al.* (MANZALINI et al., 2007) the development activities of the project are focused on prototyping a toolkit based on distributed self-similar components characterized by self-* properties. The architecture of CASCADAS is based on its Autonomic Communication Element (ACE) that is composed of: Reasoning Engine, Message Handler, Self-Model, Facilitator, Specific Part. The Self-Model is a state machine that describes the possible states of the ACE and the associated transitions. The communication to discover and self-aggregate with other ACEs is performed by the Goal Needed (GN) / Goal Achieved (GA) protocol (MANZALINI; ZAMBONELLI, 2006). One of the important features of the CASCADAS architecture is its capacity of self-organizing in order to make the ACEs interact with each other to provide the desired functionality (deployed on top of CASCADAS toolkit) in a situation-aware way without configuration efforts (MANZALINI et al., 2007).

BIOlogically inspired NETwork and Services (BIONETS) project is devoted to address pervasive computing and communication environments (MIORANDI et al., 2006). According to Carreras *et al.* (CARRERAS et al., 2007), BIONETs is inspired on living world science to deal with the problems of scale, complexity, and diversity for a rather long time. Based on this, a network looks like a living ecosystem, where services play the role of organisms, evolving and combining themselves to adapt to the environmental characteristics (*e.g.*, network topology, service dynamics). The network architecture is called Service-oriented Communication System (SOC), composed of a two-tier architecture. The first layer is called T-Nodes and is formed by cheap tiny devices (sensors, tags, etc.) that possess sensing capabilities. The second layer is called U-Node and is formed by devices running services able to produce and consume information. The T-Nodes just communicate with U-Nodes in order to answer poll messages sent by the later. On the other hand, U-Nodes communicate among themselves based on opportunistic localized P2P interactions.

Autonomic Internet (AUTOI) is an European project devoted to investigate the Future Internet, and the main challenge is change from a service agnostic Internet to a service-aware network where autonomic principles are applied for managing the resources (GALIS et al., 2009). According to Rubio-Loyola *et al.* (RUBIO-LOYOLA et al., 2009), AUTOI proposes a self-managing overlay of virtual resources that can span across heterogeneous networks. The autonomic management architecture model is composed of several distributed management systems across five abstract planes called OSKMV: Orchestration Plane (OP), Service Enablers Plane (SP), Knowledge Plane (KP), Management Plane (MP), and Virtualisation Plane (VP) (GALIS et al., 2009). According to Galis *et al.* (GALIS et al., 2008), AUTOI management system is designed to achieve the following functionalities: embedded network functions, aware and self-aware functions, adaptive and self-adaptive functions, automatic self-functions, extensibility functions, and outlay functions.

4WARD is an European project that aims at investigating a clean slate approach for the Future Internet (4WARD, 2010). One of the work packages of 4WARD project defined a new management paradigm called In-Network Management (INM) (FOLEY et al., 2008), whose main goal is to embed self-management capabilities deep inside the network nodes. According to Prieto *et al.* (PRIETO et al., 2009) the INM approach is related to autonomic computing in two ways: the management plane inside the network is self-organizing and exhibits autonomic behavior; and the functions that the management plane offers are either autonomic themselves or building blocks for autonomic management functions. The principles of the INM approach are described by Dudkowski *et al.* (DUDKOWSKI et al., 2009), and based on these principles the authors derived a functional design space composed of three-dimensional disk describing the degree of embedding, degree of autonomy, and degree of abstraction. The architecture of INM approach is described in detail by Franzke *et al.* (FRANZKE et al., 2009). Two levels were defined: communication and implementation levels. The communication level is composed of INM entities, INM Protocol (INMP), and Management Capabilities (MCs). Inside the implementation levels there are the INM Registry (INMR), Functional Components (FCs), and Management Capabilities (MCs). According to Foley *et al.* (FOLEY et al., 2008) by using the INM approach, management operations can become localized and different network elements interact based on peer-to-peer techniques.

Self-Management of Cognitive Future InterNET Elements (Self-NET) is an European project devoted to investigate the designs and prototypes of the Future Internet throughout the employment of self-management and use of cognitive functionalities (NGUENGANG et al., 2009). According to Kousaridas *et al.* (KOUSARIDAS et al., 2008) Self-NET proposes a generic cognitive cycle model that is composed of monitoring, decision, and execution process. This cycle model can be seen as a simplification of the Monitor, Analyze, Plan, Execute - Knowledge (MAPE-K) control loop proposed by IBM. Moreover, the authors proposed a distributed execution of this cycle model which happens to introduce a certain level of orchestration among the monitor, decision, and execution elements of their architecture. The details of the Self-NET architecture are described by Mihailovic *et al.* (MIHAILOVIC et al., 2009) and Nguengang *et el.* (NGUENGANG et al., 2009). A three-level environment, composed of networking, cognitive, and management levels, is described by Mihailovic *et al.* (MIHAILOVIC et al., 2009). In such three-level environment the Self-NET architecture is deployed. One of the main elements of the architecture is the Monitoring, Decision-Making, and Execution cycle (M-D-E cycle), that is responsible for conducting the self-management. The M-D-E cycle is distributed along the network element level, as Network Element Cognitive Managers (NECM), and in the network domain level, as Network Domain Cognitive Manager (NDCM). These elements form the Distributed Cognitive cycle for System & Network Management (DC-SNM). Nguengang *et al.* present the framework to provide self-awareness process for Self-NET (NGUENGANG et al., 2009).

In addition to the aforementioned projects, there are many other initiatives trying to develop autonomic networks and autonomic network management solutions, such as EFIPSANS (EFIPSANS, 2010), Autonomia (DONG et al., 2003), AutoMate (AGARWAL et al., 2003). Nevertheless, besides the investigations conducted in the context of projects, there are also independent investigations devoted to create architectural approaches for bringing together autonomic computing/self-* properties and network management. Some of these independent approaches are presented as follows.

Tizghadam and Leon-Garcia proposed the Autonomic Network Control and Management System (AORTA), which is a self-organizing mechanism to enable robustness and performance on IP packet transmission (TIZGHADAM; LEON-GARCIA, 2008). The authors based their architecture on Darwin's evolutionary concepts, and created two control loops, one for short-term and the other for long-term. The short-term address on-line problems while the long-term uses virtual networks to reconfigure the network in order do provide robustness and performance. The techniques used to provide the virtual network resource management were detailed by Farha *et al.* (FARHA; LEON-GARCIA, 2009).

Razzaque *et al.* (RAZZAQUE; DOBSON; NIXON, 2007) give another perspective to cross-layering approaches. The authors claim that in the next generation networks, which will be composed most of wireless networks, the information used for decision making should be exchanged among several layers. Based on this exchanged information each layer could adapt and optimize the end-to-end performance. Playing with local and global view the authors discuss about current cross-layer solutions and what is missing in order to provide an autonomic cross-layer architecture.

Balasubramaniam *et al.* (BALASUBRAMANIAM et al., 2006) proposed an autonomic network model based on four bio-inspired concepts: management process of the glucose, reaction diffusion, chemotaxis, and hormone signaling. The authors defined an hierarchical structure that consists of devices layers, system and business. The system

and device layers are bio-inspired, while the business is not. The device layer is inspired in three concepts, the reaction diffusion that is responsible for the self-organizing characteristic; the chemotaxis that is the ability to move micro-organisms (for example, management orders) based on the stimulus attractions which confers the self-adaptation characteristic; and finally the hormone signaling that is responsible for the communication between distant cells. The system layer is inspired on the management process of the glucose, that is a mechanism executed inside the blood and it is the basis to maintain the organism equilibrium. The behavior of the proposed autonomic network is a combination of centralized and decentralized management actions.

Despite the general concepts and high level architectures, there are several initiatives using autonomic or self-* properties directly defined to specific networks or management tasks. The next section describes some of these initiatives.

2.2.2 Specific Solutions

The analysis of the literature has presented an increasing number of proposals relating autonomic computing and self-* properties to network management. In this section, these proposals are organized into two groups: oriented to a specific network and oriented to specific management tasks.

Proposals Oriented to a Specific Network. Wireless network community is heavily using self-* properties to solve their management problems. Given the dynamic nature of wireless networks, and the inability of managing this environment in a manual fashion, the research developed for these kind of networks has been exploiting self-* and autonomic concepts. It is possible to find autonomic solutions in a very large range of wireless networks. For example, wireless sensor networks (WSN) (LU et al., 2007) (WANG; LI; ZHANG, 2007) (BOONMA; SUZUKI, 2008) (KIRI; SUGANO; MURATA, 2007), ad hoc networks (HADJIANTONIS; PAVLOU, 2008) (MALATRAS; PAVLOU, 2007) (MALATRAS; HADJIANTONIS; PAVLOU, 2007) (ZHANG; LI, 2008), next generation networks (4G, 3G, etc.) (DEMESTICHAS et al., 2007) (VIDALES et al., 2005), and pervasive networks (ZHANG; HANSEN, 2008) (CASTELLI; MENEZES; ZAMBONELLI, 2009). The details related to some the listed initiative are described below.

- A framework to build wireless sensor network applications was introduced by Boonma *et al.* (BOONMA; SUZUKI, 2008). This framework is called Multi-objective Optimization for a Network of Sensors using an evOlutionary algOrithm with coNstraints (MONSOON), and is inspired in biological concepts of bee colonies. The agents are able to evolve and adapt according to constrains. Some of the techniques employed are policies and genetic algorithms. The self-* properties provided in this framework are: self-configuration, self-optimization, and self-healing. Another work on WSN is presented by Lu *et al.* (LU et al., 2007). The authors propose a mechanism that integrates self-configuring and self-organizing features in order to manage wireless sensor networks. This work is different from many others because the authors use only local information to provide the self-organizing feature.
- Mobile ad hoc networks (MANETs) are characterized by the freedom presented by the mobile nodes. Indeed, this freedom can become a negative feature in terms of network management because the mobility of the nodes, limi-

ted connectivity, and interference turns the management impossible to be executed manually. Thus, self-* properties are a natural solution for MANETs. For example, Malatras *et al.* (MALATRAS; HADJIANTONIS; PAVLOU, 2007) used context-aware policies to adapt the system and provide self-configuration and self-optimization for MANETs. Their solution is based both in distributed and hierarchical models more suitable for this kind of network.

- The next generation of cellular communications has inherited numerous labels, including beyond 3G (B3G), enhanced 3G (E3G), and 4G. Currently, there is no formal standard or definition for 4G. While 3G systems focus primarily on supporting multimedia data rates and various classes of service, the focus of B3G and 4G systems is to seamlessly integrate existing wireless systems. All this integration process demands a considerable management effort that once again can not be handled manually. Demestichas *et al.* (DEMESTICHAS et al., 2007) presented an autonomic architecture to autonomously configure accesses points called Autonomic Management of Access Points (AMAP). This architecture shows the convergence of telecommunications and data networks in terms of network management approaches.
- The work developed by Schuetz *et al.* (SCHUETZ et al., 2007) presented an autonomic distributed solution for the management of base stations of Wireless Local Area Networks (WLAN). The authors used information retrieved from the neighbors, that reflects the network context, to feed the local decisions of the autonomic agents inside the base stations. Despite the employment of distributed algorithms, the authors also used a central element that is able to handle in a better way the management functions that require certain levels of centralization, like for example, management of the policies of the system, and the interaction with human administrators.
- In ad hoc networks, Badonnel *et al.* (BADONNEL; STATE; FESTOR, 2007) presented a probabilistic self-organizing. This approach considers distributed algorithms for self-organizing solely the nodes that are "interested" into being managed, and the decision of which node is interesting or not is defined according their probabilistic model. Karnik and Kumar developed a self-organizing model for ad hoc wireless sensor networks based on distributed algorithms (KARNIK; KUMAR, 2007). The authors' proposal is based on the analysis of local computation and communication of the sensor node.
- Considering pervasive systems, Zhang and Hansen (ZHANG; HANSEN, 2008) proposed a self-managed pervasive service middleware able to get information about the runtime status of devices, services calls, and network connections. Thus, based on changes of those status the middleware is able to dispatch self-diagnosis, self-configuration to adapt the pervasive system to the new conditions. Some of the techniques used in this work are ontologies, and context-aware information. Another work putting together pervasive systems and self-* properties is presented by Castelli *et al.* (CASTELLI; MENEZES; ZAMBONELLI, 2009). In this case, the authors focused on restricting the amount of knowledge used for the management decisions of the pervasive nodes. To achieve this, they provide a self-organizing approach to generate knowledge that is based on local information, bio-inspired algorithms, and on the similarity of required information.

Proposals Oriented to Specific Management Tasks. In the literature, there are several proposals of employing autonomic and self-* properties to design different types of management tasks (COELHO; GASPARY; TAROUCO, 2009) (GHAZALT et al., 2008)(FALLON et al., 2007). Some of these proposals are described below.

- Tizghadam and Leon-Garcia (TIZGHADAM; LEON-GARCIA, 2010) proposed AutoNet, which is a self-organizing management system for core networks where robustness to environmental changes (*e.g.*, traffic shifts, topology changes, and community of interest) is viewed as critical. The main objective is to provide an autonomic traffic engineering solution. For this, the authors defined a centralized architecture where a two-loops strategy is executed. This strategy is composed of a long-term loop and a short-term loop. The self-organizing property is part of the algorithms proposed. The essence of such algorithms is to determine the least critical paths for allocation of new traffic flow requests.
- Chaparadza *et al.* (CHAPARADZA; COSKUN; SCHIEFERDECKER, 2005a) combined self-* aspects and monitoring techniques to build a traffic self-monitoring system. The authors define that self-monitoring networks are those that autonomously decide which information should be monitored, as well as the moment and local where the monitoring task should take place.
- Yangfan Zhou and Michael Lyu (ZHOU; LYU, 2007) presented a sensor network monitoring system. The authors' contribution is the use of sensors themselves to monitor each other in addition of performing their ordinary task of sensing the surrounding environment (traditionally, the task of monitoring sensor elements is executed by external elements and not by a sensor itself).
- Viroli *et al.* (VIROLI; CASADEI; OMICINI, 2009) presented a self-organizing coordination service. The coordination service is a chemical-inspired system where elements combine themselves as if they were molecules affected by chemical laws. The authors based their chemical coordination of services solely on local criteria, and evidenced the emergence of self-organizing global coordination of the system.
- Mckinley *et al.* (MCKINLEY et al., 2006) and Samimi *et al.* (SAMIMI et al., 2007) presented the Service Clouds environment, that is an approach of putting together self-* and overlays for network management. This environment provides an infrastructure for dynamic deployment and reconfiguration of services belonging to an overlay. It presents an autonomic platform alternative to maintain the communication channels of deployed services and make them adapt according to the overlay conditions.
- Houidi *et al.* (HOUIDI; LOUATI; ZEGHLACHE, 2008) presented a distributed and autonomic mapping framework responsible for self-organizing the virtual networks on top of the substrate network every time a new deployment request arrives. This request triggers the autonomic elements, which in their turn, exchange messages to build a global view of all virtual network topologies and decide where to place/replace the resources of the virtual networks. Despite the fact this approach employs autonomic features and distribution, the self-organization is subjected solely to the changes on the number of virtual networks running on top of the substrate network. Thus, changes on

the amount of resources used by the virtual networks during their lifetime are not explicitly considered.

- Steinder *et al.* (STEINDER et al., 2007) and Wang *et al.* (WANG et al., 2008) proposed the employment of self-organizing techniques on server virtualization scenarios. In these cases, the virtual machines are self-organized according to the workloads of the physical nodes, and generally, this self-organization is accomplished migrating virtual machines to physical ones with lower workloads. The metrics traditionally used to determine the workload of virtual machines are CPU and memory.

The initiatives presented in this section reveals how much the concepts of autonomic computing and self-* properties span across network management discipline. The influence of those concepts is present from network systems with very restricted conditions (*e.g.*, wireless sensor networks) until very abstract environments (*e.g.*, virtual networks and Service Overlay Networks (SONs)). Moreover, the current amount of projects and the diversity of architectural proposals (presented in Section 2.2.1) also indicates that the research in autonomic network management is in expansion and tends to become a "de facto" alternative of network management.

2.3 Employment of P2P on Network Management

In the same way that there are no general agreement about P2P definitions, there are no standard terminology, techniques, and directives of how P2P can be employed on network management discipline. Thus, considering that there are different groups of P2P technologies, and different purposes on building P2P applications, it is possible to conclude that the employment of P2P on network management solutions can assume distinct forms. Therefore, the objective of this section is to characterize which are the current ways of developing P2P-based network management solutions. To achieve this objective, the main proposals found in the literature are analyzed and compared to the features of groups of P2P technologies and categories of P2P applications presented in Section 2.1.3. The conclusion of this comparison is that current P2P-based network management proposals can be organized into two groups. The main criterion for grouping such proposals is the common characteristics explored by the P2P approach and the type of network management tasks employed. The groups are presented as follows.

P2P infrastructure to support general purpose management platforms. This group is formed by general purpose P2P-based network management systems. The remarkable feature in this group is the employment of the P2P infrastructures (mainly routing, location, and connectivity P2P infrastructures) to enable a more flexible deployment of a network management system. Examples of initiatives in this group are presented below.

- State and Festor (STATE; FESTOR, 2003) defined a P2P infrastructure based on JXTA (GONG, 2001). The management system explores the advertising messages of JXTA as a manner of exposing the management API of the managed elements to all peers that are part of the P2P management system. In this case, the manager/agent approach is maintained, and through the P2P infrastructure the manager can access the Java Management Extensions (JMX) management agents.

- Granville *et al.* (GRANVILLE et al., 2005) employed both P2P infrastructures and P2P applications to enhance the traditional Management by Delegation (MbD) approach. P2P infrastructure is used to provide the connectivity abstractions, routing, and cross domain communications, while P2P applications, like file sharing and instant messaging, are used to support the human interaction among teams of administrators of their approach. The authors defined a MbD infrastructure composed of Top Level Managers (TLM), Middle Level Managers (MLM), and agents (PANISSON et al., 2006). TLMs are able to use the P2P applications available on the management system to enable the human collaboration. MLMs are responsible for executing the management tasks by contacting the agents. Inspired by the aforementioned work, Fiorese *et. al* (FIORESE; SIMÕES; BOAVIDA, 2009) focused their proposal on enhancing the connectivity among TLMs and MLMs by investigating the location issues of P2P infrastructures.
- Barsham *et. al* (BARSHAN; FATHY; YOUSEFI, 2009) also followed the strategy of joining P2P infrastructure with MbD. In the case of the work proposed by these authors, a 3-tier hierarchy of peers is composed of TLM, MLM, and Lower level Managers (LLM). The focus of their research is to provide fault tolerance for a P2P-based MbD approach.
- Kamienski *et. al* (KAMIENSKI et al., 2006) used P2P infrastructure to provide a better support on the management of policies. The authors kept the same hierarchical concept behind the Policy-based Network Management approach. However, instead of using a Policy Decision Point (PDP), they used Policy Decision Nodes interconnected by a Distributed Hash Table (DHT) network. Through the DHT it was possible to reach and change the policies. In this sense, the P2P infrastructure is used to enable the dissemination of policies inside the management system. The authors aim at providing scalability and fault tolerance for the network management system.

P2P infrastructure to provide specific management solutions. This group comprises the initiatives where P2P is used to solve some specific management tasks and situations. Examples of such initiatives are listed below.

- Yalagandula *et. al* (YALAGANDULA et al., 2006) designed a sensing information management backplane that, among other techniques and technologies, employed DHTs algorithms to aggregate and disseminate network and node status information. Web services are used to enable the composition and aggregation of monitored information, and a P2P infrastructure created by the DHTs and overlays is used to expose the WS interfaces and the monitored information. In this context, P2P is used as the main infrastructure to support scalable dissemination of monitored information, and location of the services of the management system.
- Zhou and van Renesse (ZHOU; RENESSE, 2005) employed a structured P2P infrastructure (by using DHT algorithms) for helping on the maintenance of connectivity information about IPv6 and IPv4 networks. The authors established that the core network will keep being IPv4, while the edges will be IPv6 networks. To solve the connectivity problem, the authors proposed that egress gateways from IPv4 networks use DHTs in order to keep routing tables with

information able to handle the mapping between the two types of networks. Leng *et. al* (LENG et al., 2007) also proposed a P2P infrastructure to address the connectivity problem between IPv4 and IPv6 networks. In this work an unstructured P2P network was used to distribute Tunnel End Point (TEP) information among the IPv4 gateways. Analyzing both proposals, in the light of P2P technologies, it is possible to say that the authors used P2P infrastructures in order to build a content distribution application to disseminate routing information and enhance the connectivity between end points.

The predominant feature on the proposals analyzed above is the employment of P2P infrastructures to enhance the underlying conditions of the network management systems. Very few initiatives use the concepts behind the P2P applications in order to enhance the execution of the network management task itself. During the analysis of the works described above, it was recurrently mentioned the cooperation among the peers of the P2P-based network management infrastructure. However, it is was never clear what exactly the authors meant with the term cooperation. In most of the cases, this term indicated a connectivity relation between the peers rather than a joint operation to solve a problem. Indeed, the literature shows that P2P infrastructures are being well explored to build network management infrastructures, while management applications keep on being developed following traditional hierarchical network management approaches.

2.4 Autonomic/Self-*, Peer-to-Peer, and Network Management

As presented in the previous sections, there are several proposals joining autonomic or self-* properties and network management, as well as, P2P and network management. Nevertheless, when all these topics (*i.e.*, autonomic/self-*, P2P, and network management) are put together it is possible to observe that there are no clear proposals fully merging those topics. For example, Prieto *et al.* (PRIETO et al., 2009) and Franzke *et al.* (FRANZKE et al., 2009) mentioned that nodes of INM approach (in 4WARD Project) would follow a P2P interaction model. However, no precise details and definition were described relating self-* properties and P2P for executing the in-network management defended by the authors. The same lack of clear definition, related to the employment of the three topics, happens to the proposal presented in BIONETs project (MIORANDI et al., 2006) (CARRERAS et al., 2007). Some of the few initiatives joining autonomic/self-* and P2P techniques for network management are described and analyzed as follows.

Binzenhöfer *et al.* (BINZENHÖFER et al., 2006) employed P2P overlays to address fault and performance management. Their architecture aims at providing generic connectivity tests and Quality of Service (QoS) monitoring in a distributed and self-organized system that is based on the Distributed Network Agents (DNAs) (JUN et al., 2007). The distributed infrastructure is achieved by the employment of overlays formed by structured P2P networks (using DHTs) on top of the monitored network. In this sense, groups of DNAs composing a DHT are able to communicate to: exchange monitoring information; and ask for other DNAs to execute tests on the monitored network in order to find eventual failures. The self-organizing property is related to the maintenance of the distinct overlays that might be defined during the execution of this environment (JUN et al., 2007). In this sense, the self-* property is not directly related to the network management task being executed

(*i.e.*, monitoring and QoS tests) but it is related to the maintenance of the P2P overlays. In addition, the authors described that the decision of which peer will belong to a monitoring/testing overlay is given by random choice or by human definition (BINZENHÖFER et al., 2006). This description emphasizes that the management task being executed does not reveal a truly self-organizing behavior. An example of self-organizing behavior would be the management P2P overlay itself discovered which are the suitable peers to form and execute a monitoring or test request.

Brunner *et. al* (BRUNNER et al., 2005) proposed the Ambient Network (AN) concept that is based on the composition of different networks in order to gain connectivity. The authors suggested that a P2P-based network management approach could be able to handle the network compositions of the AN concept in two manners. One manner is related to the topological composition between management systems of ANs, and the second is associated to the creation of the connectivity conditions required to compose two ANs. In this case, P2P technology is used for the maintenance of the hierarchical management overlay, and for pooling and sharing management information within and across heterogeneous composed networks. Simon *et. al* (SIMON et al., 2005) detailed the employment of P2P approach to enable the composition of ANs. It is not discussed on both works how the management tasks running inside the composed networks should exactly work. Apparently, the management tasks would be executed in an hierarchical fashion, where super peers (*i.e.*, managers) request for peers (*i.e.*, agents) the execution of some task. So, P2P technology is used to support the connectivity across domains, provide scalability of the network management system, and disseminate information. Besides the employment of P2P, self-management is also incorporated in Ambient Networks. Mathieu *et al.* (MATHIEU et al., 2007) proposed the self-management of contexts associated to the overlays of AN. The authors defined the Service-aware Adaptive Transport Overlays (SATOs) for ANs. A SATO is created for delivering a certain requested service. The self-management of SATOs is accomplished throughout the collection of distributed context associated to users and networks, and the assignment of dedicated nodes to analyze the collected information. Based on this information, SATOs can be deployed and adapted. Analyzing the works related to AN, above listed, it is possible to identify that there is not a clear and direct connection between P2P and self-management devoted to constitute a management solution for the Ambient Network concept. The presence of P2P is very strong on the management of ANs, however, self-management is more related to the users perspective, rather than to the management of ANs.

Fallon *et. al* (FALLON et al., 2007) employs a P2P approach to self-form network management topologies targeted to accomplish specific network management tasks. A hierarchical model based on Network Elements (NE) is employed. The NEs are grouped into clusters, and these clusters form P2P overlays that can be arranged hierarchically according to the requirements of the management task to be executed. One conclusion that is possible to be inferred from this work is the fact that the cooperation among the NEs performing a management task is not provided by their proposal, solely the connectivity of such NEs is provided (*i.e.*, the arrangement of the NEs in an overlay). The self-forming property is associated to the process of preparing the network management infrastructure. Based on parameters associated to the NEs, the clusters are formed, maintained, and self-optimized in the presence

of changes. The parameters can be changed dynamically by direct operator intervention, automatically using policies, and because of changes on the network status (FALLON et al., 2007). Analyzing the proposed work it is easy to distinguish the relationship between the self-* properties and the P2P management overlay. In the same way, it is possible to identify the integration between the management overlay and the execution of a management task. However, it is not easy to understand the influence of the self-* properties on the execution of the network management task. Indeed, at a first glance, the network management task, considered by the authors, is modeled taking into account the cooperation provided by the P2P management overlay, but no self-* properties are considered to be part of the network management task.

Besides the initiatives presented here, there are other proposals concerning the joint use of self-* and P2P and/or overlay (AL-OQILY; KARMOUCH, 2008) (WANG et al., 2006). However, those proposals are not directly related to network management, but they address the management of the P2P and/or overlay network itself. Therefore, this thesis proposal aims at bringing knowledge to issues involving the joint use of self-* properties and P2P to contribute with the development of an alternative for designing network management task solutions.

2.5 Summary

This chapter presented an overview of the state of the art regarding network management, autonomic computing, peer-to-peer, and the relationship among these three research areas. First, the discussion about network management approaches available in the literature was presented. Analyzing the discussion and the output of them it was possible to verify the majority presence of distributed network management approaches, and the indication by the NMRG-IRTF that more investigations are necessary on this area. Indeed, the research community understands, as a common sense, that distributed solutions are more suitable to handle the current scenarios where network management is employed. In the sequence, a review of the definitions associated with autonomic computing and the current research status of P2P were presented. Some of the main architectural approaches for autonomic and network management were presented, and also some individual initiatives to develop self-* properties applied to network management. The diversity of contexts and areas is one of the characteristics of research on autonomic network management. Considering the P2P scope, this chapter presented the types of investigations and how they are related to network management, and once again, the broadness of the solutions is a remarkable feature. This chapter is closed with the discussion about the proposals related to: autonomic or self-* properties applied to network management; P2P employment on network management; and finally the combination of autonomic or self-* properties and P2P on network management.

3 PRINCIPLES OF THE THESIS

This chapter starts with the description of the conditions on the network management community that lead to the proposal of the joint use of self-* properties and P2P. Next, it is described the features of the network environments that could benefit from a self-* P2P network management alternative, and it is also defined the management requirements associated to those environments and the self-* P2P alternative.

Due to the fact that the literature shows different point of views related to self-* properties, autonomic computing, peer-to-peer, and cooperation, there is one section establishing which are the meanings of the aforementioned concepts in the context of this thesis, and what are their relationship to the self-* P2P alternative. In addition, design issues are proposed to be followed on the integration of self-* properties and P2P so that it becomes possible to design fully distributed and cooperative network management applications.

Finally, the methodology employed to verify the hypothesis of this thesis relies on the investigation of scenarios where self-* properties and P2P can be used together in order to improve a network management solution. Case studies are used to carry on the investigation, and they are presented at the end of this chapter.

3.1 Leading Conditions towards the Self-* P2P Alternative

In 2006, during a meeting of the NMRG-IRTF group, the network management community identified some major problems preventing the development of new alternatives to handle the current challenges of the field (*e.g.*, reduction of human intervention, heterogeneity, scalability, reliability). From the listed problems, two of them were particularly cited by different members of the meeting (SCHÖNWÄLDER et al., 2006). One is the lack of investments on management applications and the other is the lack of heavy development on fully distributed and cooperative solutions. Indeed, it is possible to say that those two problems are interconnected.

Generally, the development of network and services management proposals keeps the focus on morphological aspects, such as APIs, protocols, architectures, and frameworks. There is almost no focus on developing the management application itself, *i.e.* the algorithms employed to execute the management tasks. Thus, the management applications are in most of the cases limited by the morphological aspects. For example, network management applications developed to use SNMP protocol are very simple. The algorithm behind an SNMP-based management application follows a master-slave approach, where the master (*i.e.*, the manager) sends tasks (*i.e.*, gets, sets, getbulk, etc.) to the slave (*i.e*, agent) and waits for a response. Therefore, the limitations imposed by the morphological aspects generally turn the management solution into centralized (*i.e.*, manager-agent based approach) or not fully distributed architectures (such as hierarchical approach).

In the last years, there were attempts to define architectures and frameworks able to provide fully distributed and cooperative network management infrastructures, but those attempts did not provide truly cooperative applications. Examples of such attempts are the solutions based on P2P and MAS (Multi Agent Systems). As presented in Section 2.3, P2P technologies are employed in the sense of improving the connectivity capabilities of the management infrastructure, so that cooperative applications could be developed. Nevertheless, the network management applications designed so far considering P2P technologies are based on hierarchical approach to execute the management tasks (*e.g.*, MbD, PBNM). The same way, MAS solutions enable a more distributed network management infrastructure, where agents (or intelligent agents) are spread along the network to be managed. Most of the proposals of MAS-based network management solutions claim that the agents cooperate by sub-tasking/splitting/replicating the management task among multiple agents, or by gathering information from other agents for individual deliberation of management actions (LI et al., 2001) (AKASHI et al., 2005) (TERAUCHI; AKASHI, 2009) (GUARDALBEN et al., 2010). This way, most of the P2P and MAS-based network management applications still present a master-slave behavior, which is a simple and tending to be centralized style of defining an application.

Parallel to the attempts of developing more distributed and cooperative management infrastructures, sophisticated network management algorithms started to be designed with the introduction of self-* properties and autonomic computing into the research community. Most of the solutions following the first definition of autonomic computing (KEPHART; CHESS, 2003) tend to employ centralized algorithms on the design of the autonomic decisions. This tendency is justified due to the fact that the authors of this first definition did not consider the necessity of communication among the autonomic managers. Thus, centralized managers are build incorporating the autonomic functionalities while agents keep feeding those managers with information or executing the demanded tasks. In contrast, autonomic solutions based on multi-agent systems emerged as a solution to build distributed autonomic applications. However, autonomic initiatives based on multi-agent systems suffer from the problem of guaranteeing that the behavior emerging from individual goals of each agent will truly result in the common goal to be achieved by the system (HUEBSCHER; MCCANN, 2008). The alternative to multi-agent systems is to deploy hierarchies of autonomic managers, which in most of the cases keep the MbD approach of developing the network management applications.

As the literature shows, the initiatives so far proposed were not fully capable of developing side by side sophisticated network management applications and fully distributed and cooperative solutions. In fact, to achieve a highly distributed and cooperative solution it becomes necessary more investments on the application that is running on the distributed management entities. Therefore, the investigations conducted in this thesis are devoted to explore different angles and aspects from those so far employed (*i.e.*, morphological aspects). In this thesis, it is of special interest explore how network management applications can be designed when sophisticated techniques like self-*properties are combined with the distributed nature of P2P technologies. The goal is to investigate the connectivity and cooperative capabilities of P2P as the anchor for real distribution and cooperation on the execution of self-* network management applications. The foundations of such investigation are described in the next sections.

3.2 Characterization of Networks and Management Requirements

The range of network environments that could be investigated is very large. There are, for example, the traditional Local Area Network (LAN) (TRIPATHI; HUANG; JAJODIA, 1987), Metropolitan Area Network (MAN) (KABATEPE; VASTOLA, 1996), Wide Area Network (WAN) (CAVENDISH, 2004), and the emerging ones like Wireless LAN (WLAN) (LIN; CHENG, 2005), Worldwide Interoperability for Microwave Access (WiMAX) (SEKERCIOGLU; IVANOVICH; YEGIN, 2009), Vehicular Ad Hoc Network (VANET) (HAAS; HU; LABERTEAUX, 2009), Personal Area Network (PAN) (CAMPOS; RICARDO, 2006). The analysis of Chapter 2 shows that most of the networks related to wireless technologies typically use some kind of self-* property and distributed methods for building up the communication infrastructure and management of the network. The main reason for the employment of these methods is related to the proper nature of wireless networks. In general, such networks are based on a large number of components, dynamic changes on the surrounding environment, mobility, etc. Instability is the basis of wireless networks, and relying on humans or centralized elements to provide the control infrastructure for this environment is not the most appropriated option.

Different from wireless networks, the nature of wired networks tends to be more stable, which reduces the complexity of management solutions. Indeed, the literature shows that most of the network management alternatives proposed to wired networks relied on centralized or hierarchical solutions that are easier to be deployed and maintained. However, challenges have been posed to the traditional management solutions on wired networks. Some of these challenges are: the increasing size of the networks, online multimedia applications, and the integration of the traffic from wireless networks into the wired networks (users are connected via wireless technology to the Internet at any time and place). As a consequence, the management of wired networks has been pushed to cope with dynamic changes, and instability on the quality of service, and user's quality of experience in such a scale that was never seen before, and that tends to increase even more. In this sense, traditional and conservative management solutions have their employment questioned, while sophisticated methods, such as self-* properties and P2P-based management, are pointed as good solutions for the management of wired networks.

In fact, despite the nature of the network (wireless or wired), dynamic changes and the need of reducing human intervention are some of the major characteristics of current networks. In this thesis, it is believed that the presence of those characteristics implies a set of specific management requirements, and this set justify the development of self-* P2P based network management applications. Thus, the set of management requirements considered in this thesis is defined and presented below.

Efficient use of the resources. The management application running inside the network can not impose a load that compromises the operation of the proper network in terms of traffic or processing power of the management or managed entities.

Agile management actions. This requirement is associated with the reduction of manual intervention. Depending on the size of the network environment to be managed, or on the dynamics on the changes of the user's requirements, it becomes impossible to effectively perform management tasks in a centralized or manual fashion. Thus, the objective is to turn manual tasks into tasks executed by the management application itself. Examples of manual tasks are: identification of saturated resources, verification of the status of management entities, managed devices, alarms, etc.

Management actions must be transparent for the users utilizing the resources of the network. The investigation carried on this thesis is interested in the development of self-* network management applications that can follow a fully distributed and cooperative approach. One important issue on distributed computing is the transparency of the solution. According to A.S. Tanenbaum and M. Van Steen there are different types of transparency (access, location, migration, relocation, replication, concurrency, failure) (TANENBAUM; STEEN, 2007). The management task must be designed to support dynamic changes on the environment and at the same time provide the suitable types of transparency.

Enable more parallel and simultaneous behavior on the management actions instead of sequential one. Most of the network and services management applications are based on centralized and hierarchical models. Those models create a chain of sequential steps that must be followed by the management entities. This chain prevent the parallel and simultaneous execution of the management task in different parts of the managed network, which can become a problem. A common solution for this problem is to increase the number of managers in different parts of the network (making even more strong the hierarchical model), creating a "vertical parallelization" that should reproduce a parallel and simultaneous behavior. In this thesis, however, it is believed that this kind of "vertical parallelization" is not suitable anymore. Thus, it is necessary to enable a "horizontal parallelization" on the behavior of management applications.

The aforementioned management requirements are important in the context of this thesis because they help to restrict the network environments that can help on the investigation of self-*, P2P, and cooperative solutions. Nonetheless, there can exist other management requirement that can also be consider in the investigation of self-* P2P employment, like for example, security. The key is that network environments matching the established set of management requirements demand more sophisticated management solutions, like the one proposed to be studied in this thesis. The details of the investigation on the joint use of self-* properties and P2P are described in the next sections.

3.3 Definition and Delimitation of Terms and Concepts

The literature shows different points of view for the terms related to self-* properties, autonomic computing, peer-to-peer, and cooperation. For this reason, this section depicts the definitions of the aforementioned terms used in the context of this thesis. Moreover, it is presented a delimitation of which terms are of special interest.

Self-element. This term is defined here in this thesis. It represents a management entity that executes some kind of operation that contributes on providing a self-* property. For this thesis, a self-* property is not necessary mapped to a self-element, but this property could be composed of different self-elements.

Self-* property. In the literature, there is not a proper definition for a self-* property. Therefore, here, this term reflects a certain action that is designed to be executed with minor human intervention. Based on this definition, it is possible to match the properties already established, like self-healing, self-configuration, self-optimizing, self-protection, and also define others such as self-organizing, self-awareness, self-monitoring.

Self-management. This term is also not precisely defined, and as emphasized by Huebscher and McCann (HUEBSCHER; MCCANN, 2008) there remains debates about what self-management really is. For the context of this thesis, this term regards to a system capability of being constitute of one or more self-* properties.

Autonomic. After several years of discussion, a consensus on what means the term autonomic seems to be achieved. This consensus defines autonomic computing, system, or behavior as a self-managed system presenting the self-CHOP properties, *i.e.*, self-configuring, -healing, - optimizing, and -protection (SAMAAN; KARMOUCH, 2009) (HUEBSCHER; MCCANN, 2008).

Autonomic Network Management System (ANMS). According to Samaan and Karmouch (SAMAAN; KARMOUCH, 2009), an ANMS is a network management system that employs the autonomic computing concept. Thus, an ANMS must perform management operations following the self-CHOP properties.

Autonomic Element (AE) and Autonomic Manager (AM). One common accepted definition of autonomic element was given by Kephart and Chess (KEPHART; CHESS, 2003): *"Individual system constituents that contain resources and deliver services to humans and other autonomic elements. Autonomic elements will manage their internal behavior and their relationships with other autonomic elements in accordance with policies that humans or other elements have established."*. This definition is known in the literature as the IBM autonomic element. The classic representation of an IBM autonomic element is illustrated in Figure 3.1. Analyzing this illustration it is possible to see that the IBM AE is composed of managed element and autonomic manager. A more specific definition of autonomic manager (considering the IBM AE architecture) was provided by Huebscher and McCann (HUEBSCHER; MCCANN, 2008), where an autonomic manager *"is a software component that ideally can be configured by human administrators using high-level goals and uses the monitored data from sensors and internal knowledge of the system to plan and execute, based on these high-level goals, the low-level actions that are necessary to achieve these goals."*. Although the IBM autonomic element is the most adopted definition and architecture, there are other attempts to define autonomic managers and elements (TRUMLER et al., 2005) (MEER et al., 2006) (MANZALINI; ZAMBONELLI, 2006).

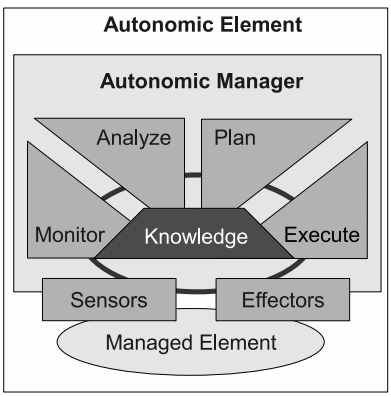


Figure 3.1: IBM autonomic element (HUEBSCHER; MCCANN, 2008)

Peer-To-Peer. This thesis uses the definition of Milojicic *et al.* (MILOJICIC et al., 2002). According to those authors, P2P regards to *"a class of systems and applications that employ distributed resources in order to execute critical functions in a decentralized fashion"*. The main distributed resources are: computational power, data, bandwidth, devices, people, among others. The critical functions are related to sharing data, and the communication or collaboration among systems. The decentralized fashion associated to P2P systems is achieved throughout algorithms and data used by those systems.

Cooperation. There are many definitions for this term in the literature. In the context of this thesis, the definition of cooperation is aligned to the one described by Zomaya where both sides cooperating know each other mutually and fulfill complementary functions to serve a common objective or interest (ZOMAYA, 1996). In addition, according to Jacques Ferber there are different forms of cooperation such as an intentional posture, cooperation from the observer's point of view, increasing survival capacity, improving performance, conflict resolution (FERBER, 1999). In this thesis, the cooperation form investigated is the intentional posture. Moreover, Jacques Ferber also identified methods of cooperation, being them: (1) grouping and multiplication, (2) communication, (3) specialization, (4) collaborating by sharing tasks and resources, (5) coordination of actions, and (6)conflict resolution by arbitration and negotiation. The design of self-* P2P solutions will make use of some these methods of cooperation (more specifically 1-5) in order to provide truly distributed and cooperative network management applications.

It is important to remark that this thesis does not address the design and investigation of autonomic network management systems. For this reason, the investigation conducted here is not interested in examining self-CHOP properties for network management. In contrast, the goal of this thesis is to investigate how self-* properties can be combined with P2P in order to provide truly distributed and cooperative network management applications. Thus, based on the aforementioned definitions, the investigation of self-* P2P solutions is also related to self-management solutions for networks. It is also important to remark that the terms self-* or self-management are used in this thesis referring to the capacity that a system presents to execute actions by itself, without the human intervention. This way, the employment and development of artificial intelligent techniques (*e.g.*, reasoning) applied to network management is out of the scope of this thesis.

In addition, to keep the alignment among the definitions of the terms and the goal of this thesis, the term *Self-element* is defined in this thesis in order to make an explicit differentiation from the terms *Autonomic Element* and *Autonomic Manager*. As illustrated in Figure 3.1, the most known definitions of AE and AM present a very centralized nature, because the components that give the autonomic characteristics (MAPE-K) are placed inside a single element. In contrast, the self-element is conceived to fit into a distributed infrastructure such as the one provided by the P2P technologies, where the necessary tasks to provide a self-* property can be spread in different peers. In order to design the self-* solutions it has become necessary to define integration issues. The next section presents and discusses those integration issues.

3.4 Proposal of Integration Issues for Designing the Self-* P2P Alternative

The hypothesis of this thesis is grounded on the fact that the combination of self-* properties and P2P enhances the development and execution of network management solutions. Thus, the definition of issues to guide the integration of these techniques (*i.e.*, self-* properties and P2P) is of imperative importance, because based on these issues, models, architectures, and algorithms will be developed. Therefore, the objective of this section is to describe the issues to be addressed by the integration of self-* properties and P2P, here called self-* P2P based approach for network management. The main issues to support the self-* P2P approach are listed below.

Common knowledge of the management task. Traditional network management approaches rely on manager-agent strategy (where managers have the management task knowledge and the agents execute orders), delegated scripts (where the management task knowledge is transfered from one to another management entity), mobile code (that encloses the entire knowledge of the management task inside one management entity), and remote invocation (where the knowledge of the management task is compartmented and disjoint among several entities). In all these traditional cases, the management entities are heterogeneous, and the knowledge of the management task and how it has to be executed is not completely clear for the entities of the managed network. Therefore, conflicts, interferences, and several communications among them happen in order to execute the management tasks and get the whole idea of what is related to this task. In a self-* P2P based design, the management entities should be homogeneous, *i.e.*, they all are designed to know the entire knowledge of the management task and they have the same capabilities.

Local information. The management of complex, dynamic, and heterogeneous scenarios requires agile actions. The use of local information to base these actions can reduce the amount of time and resources used for gathering information from other elements on the network. The challenge of using local information resides on identifying which information can provide the evidences to support management actions.

Parallel and distributed algorithms for decision-making. The effects associated to the execution of network management tasks have an impact not only in a single network device or service, but they might affect segments or even the overall behavior of the network. Thus, it is reasonable to consider that the decision of executing a management task should be designed in a parallel and distributed way so that the enforcement of the management task could have a certain level of approval from other management entities of the network.

Light self-elements. The main idea is to keep this self-elements as less complex as possible, *i.e.*, keep them light and specialized. The complexity of a self-* property should reside on the logic of combining these self-elements, and not on the logic of each self-element.

Reduction of explicit and global coordination. This issue is a mechanism to guide the joint use of all the other ones mentioned before. There is a potential risk that the previous issues could lead to the definition of a self-* P2P design that requires

a considerable amount of information from neighbors and high levels of coordination among them. It is important to keep in mind that the less communication and coordination are employed the better are the chances to enhance the execution of a network management task in dynamic and heterogeneous environments. Thus, the use of common knowledge of the management task, local information, parallel and distributed decision making, and light self-elements has to result in less explicit coordination and global communications.

Nevertheless, it is possible that the combination of all those principles will not always be perfectly achieved. To measure how those issues are accomplished in the case studies, a scale with three degrees (self-explained) is defined:

- *achieved*;
- *partially achieved*;
- and *not achieved*.

The self-* P2P solutions developed for each case study will be evaluated comparing the description of the issues to be solved with the degree of their achievement on such solutions. In fact, there are trade-offs that need to be closely examined when the development of the issues defended in this thesis. These trade-offs are investigated in each case study.

3.5 Employment of Integration Issues on the Case Studies

The number of case studies is proportional to the number of self-* properties investigated, therefore, two case studies are exploited. Actually, the definition of each case study depends on the identification of a network environment, whose characteristics match the management requirements established in this thesis. The first one is related to the investigation of self-healing and P2P applied to the fault management of monitoring systems. The second case study explores self-organizing and P2P interactions in order to provide performance management of substrate resources in network virtualization environments. The detailed description of the case studies and the motivation for the investigation of the joint use of self-property and P2P in each case study are depicted as follows.

3.5.1 CS I: Self-Healing Monitoring Systems

Network and service monitoring is an essential activity to identify problems in underlying communication infrastructures of modern organizations. Monitoring is typically materialized by systems that periodically contact elements (*e.g.*, network devices and services) to check their availability and internal status. A monitoring system may be simple like the Multi Router Traffic Grapher (MRTG) (OETIKER, 1998) or complex, being composed of as diverse entities as monitors, agents, and event notifiers. The information collected and processed by monitoring systems enables human administrators, responsible for managing the Information Technology (IT) infrastructure, to identify problems, and thus react in order to keep the managed infrastructure operating in a proper way.

Monitoring systems must run uninterruptedly to ensure that failures in the managed elements are detected. Problems in the monitoring systems break the monitoring process, and can lead the human administrator to believe that the managed elements are working

properly even when they are not. Robust monitoring systems should, thus, employ mechanisms not only to identify failures on the managed infrastructure, but also to recover the faulty monitoring solution itself. However, most monitoring systems force the administrator to manually recover the occasionally broken solution. Such a manual approach may not drastically affect the monitoring of small networks, but in larger infrastructures this approach will not scale and should be replaced by efficient alternatives.

The self-managed approach is one alternative emerging as a solution for the manual approach. Typically, a self-managed system is built on top of self-* features capable of reducing the human intervention and providing more efficient results. Nevertheless, distributed monitoring systems have been explored as the most popular alternative (TRIMINTZIOS et al., 2006) (AWERBUCH; KHANDEKAR, 2007) (CHOURMOUZIADIS; DUQUE; PAVLOU, 2009). Most of this popular proposals define complex monitoring systems that generally identify internal failures and employ algorithms to reorganize itself without the failed components. In this sense, these proposals present a certain level of self-awareness and adaptation, although self-healing is not present, *i.e.*, failing entities are not recovered or replaced. It means that in scenarios where most of the monitoring entities crash, the monitoring systems stop working because no mechanism is employed to maintain the execution of the monitoring entities.

Yalagandula *et al.* (YALAGANDULA et al., 2006) propose an architecture for monitoring large networks based on sensors, sensing information backplane, and scalable inference engine. The communication among the entities relies on a P2P management overlay using DHTs. Prieto and Stadler (PRIETO; STADLER, 2007) introduce a monitoring protocol that uses spanning trees to rebuild the P2P overlay used for the communications among the nodes of the monitoring system. Both Yalagandula *et al.* and Prieto and Stadler works can reorganize the monitoring infrastructure if failures are detected in monitoring nodes. After such reorganization, the failing nodes are excluded from the core of the rebuilt monitoring infrastructure. Although reorganized, with fewer nodes, the monitoring capacity of the system is reduced as a whole. Again, adaptation is present, but proper self-healing is not.

In fact, few investigations explicitly employ self-managed concepts in monitoring systems. Chaparadza *et al.* (CHAPARADZA; COSKUN; SCHIEFERDECKER, 2005b), for example, combine self-* aspects and monitoring techniques to build a traffic self-monitoring system. The authors define that self-monitoring networks are those that autonomously decide which information should be monitored, as well as the moment and local where the monitoring task should take place. Nevertheless, such work does not define how the monitoring system should react in case of failures in its components. The meaning of self-monitoring in this case is different than the one employed in this case study. While self-monitoring in Chaparadza's work means autonomous decision about the monitoring process, in this thesis, self-monitoring is about detecting problems, through monitoring techniques, in the monitoring system itself.

Yangfan Zhou and Michael Lyu (ZHOU; LYU, 2007) present a sensor network monitoring system closer to the view here adopted. The contribution of this work is the use of sensors themselves to monitor one another in addition to performing their original task of sensing their surrounding environment. Although self-monitoring is achieved, given the restrictions of the sensor nodes (*e.g.* limited lifetime due to low-capacity batteries), the system cannot heal itself by reactivating dead nodes.

Based on the aforementioned, it is possible to verify that new approaches for self-monitoring systems are required. New proposals should explicitly include, in addition to

self-awareness already available in the current investigations, self-healing support on the monitoring entities in order to autonomously keep the monitoring service up. Therefore, this thesis investigates the joint use of the self-healing property with P2P techniques in order to define a robust service monitoring system.

3.5.2 CS II: Self-Organizing Resources on Network Virtualization

Autonomic communications are a suitable approach to deal with complex and dynamic networks. The key concept behind this approach is of building sophisticated networks capable of managing themselves in order to deal with changes from the surrounding environment. Among the initiatives employing autonomic communications, the ones related to virtual technologies deserve special attention due to their complexity, dynamics, and potential to be economically exploited. Network virtualization is an example of virtual technology that is emerging as a promising cost-effective solution for future network deployments (CHOWDHURY; BOUTABA, 2009).

Network virtualization differs from current virtual machine and virtual network approaches. The difference relies on the type of resources that are virtualized. Virtual machine employs multiplexing techniques to virtualize CPU, memory, storages, and device interfaces (EGI et al., 2007). Virtual networks multiplex physical links and build paths connecting edge customers (OHSITA et al., 2007). In this case, the network elements connecting the edge customers (*e.g.*, routers, switches, etc.) are not perceived as elements of the customer network, but they form a "tunnel" connecting edges. Finally, network virtualization multiplexes all substrate network resources (*i.e.*, physical links, routers, servers, base stations) (WANG et al., 2008). The end environment is a set of slices of substrate resources that forms an entire new network deployed on top of a physical infrastructure. Indeed, this set enables the creation of a virtual network capable of running its own protocols, routing process, services, and management solutions.

One of the major benefits of network virtualization is to outsource the operational costs associated with physical infrastructures to a single provider. For example, multimedia providers may deploy their services, like IP Television (IPTV) services, without dealing with high investments on the physical infrastructure (DEGRANDE et al., 2008) (HAN; LISLE; NEHIB, 2008). As a complement, a physical or substrate provider could multiplex its physical network to enforce multiples multimedia providers. The resource management of traditional physical networks demands a lot of effort. So, it is reasonable to think that resource management on network virtualization requires even more efforts.

For instance, consider the problem where two distinct virtual networks require conflicting amount of resources in the same substrate network area. If this problem is detected during the deployment phase, the substrate provider can employ traditional techniques, like traffic matrix optimization and load balancing, and achieve a successful usage of the substrate resources after the deployment of the virtual networks. However, if this problem occurs during the lifetime of the virtual networks it becomes necessary to make dynamic and on-line changes on the environment. In this case, the employment of traditional techniques present limitations because in general they use centralized, total-view, and off-line approaches to manage the network resources (OHSITA et al., 2007) (MIYAMURA et al., 2008). The major limitations are low responsiveness to network changes, overhead introduced by the management traffic related to the central entity, and high latency of analysis and enforcement of changes.

In addition, most of the current research in this area focuses on defining an efficient mapping or embedding process of virtual networks into the substrate network, while there

are almost no efforts focused on managing the resources during the lifetime of the virtual networks. For example, Houidi *et al.* (HOUIDI; LOUATI; ZEGHLACHE, 2008) presented a distributed and autonomic mapping framework responsible for self-organizing the virtual networks on top of the substrate network every time a new deployment request arrives. In this work, the self-organization is subjected solely to the changes on the number of virtual networks running on top of the substrate network. Thus, changes on the amount of resources used by the virtual networks during their lifetime are not explicitly considered.

The work presented by Yuy *et al.* (YU et al., 2008) deals with dynamic requests for embedding and removing virtual networks. The authors map the constrains of the virtual network to the substrate network by splitting the requirements of one virtual link in more than one substrate link. A time window is used to regulate when a reorganization of the virtual links is required. The problem of this approach is to define a time window also able to deal with changes on the use of the resource during the lifetime of the virtual networks, and not only with the dynamics on embedding and removing virtual networks. Chowdhury *et al.* (CHOWDHURY; RAHMAN; BOUTABA, 2009) proposed algorithms for embedding a virtual network that correlates both node and link mapping requirements. The process is divided in two phases. First, virtual nodes are mapped and then takes place the mapping process of the virtual links. Again, this approach deals with the deployment phase, but does not address changes during the lifetime of virtual networks.

In this sense, more sophisticated management techniques are demanded in order to cope with changes on the amount of substrate resources used by the virtual networks during their lifetime. The techniques offered by self-management research rise as an appropriated alternative to address the challenges of maintaining the efficient use of substrate resources on network virtualization, while P2P can handle the decentralization typically found in virtual network investigations. Thus, this thesis investigates the joint use of self-organizing and P2P to manage the substrate network resources. This model is based on parallel and distributed algorithms running inside each substrate node, and thus dismissing any kind of central entity. Based on local information, and direct interaction with peer neighbors, the substrate node decides to self-organize the substrate network in order to cope with the changes on traffic loads of the virtual networks. The decision of when a self-organization is required is subjected solely to resource consumption conditions of the running substrate network infrastructure. External interferences, like a new virtual network embedding or removing, are not the main issues to be managed, but the effects of any kind of change on the substrate environment is the target of the proposed approach.

3.6 Summary

This chapter describes in a first moment the conditions on the network management community that lead to the proposal of the joint use of self-* properties and P2P. Once the conditions are described, this chapter presents the management requirements of network environments that could use self-* P2P solutions. In this thesis, four management requirements were identified: (i) efficient use of the resources, (ii) agile management actions, (iii) transparency of management actions; (iv) and more parallel and simultaneous behavior on the management actions. Due to the existence of several meanings for terms related to the self-* P2P approach, this chapter also depicts the definition and delimitation of which terms and concepts are important for this thesis, as well as their understanding. To complement the basis of the self-* P2P approach, this chapter shows the proposal of

integration issues on the design of self-* P2P solutions for network management. In fact, the investigations of how should be the integration of self-* properties and P2P are regulated by the issues established to be treated in this thesis. There are five main issues to be chased: (i) common knowledge of the management task, (ii) local information, (iii) parallel and distributed algorithms for decision-making; (iv) light self-elements, (v) and reduction of explicit and global coordination. Thus, two case studies were identified as network environments requiring management solutions that fit with the proposed self-* P2P approach. The first one regards the investigation of P2P and self-healing for fault management of monitoring systems. The second case study investigates the P2P interactions associated with the self-organizing property in a network virtualization environment. The next chapters present the case studies, depicting models, implementation, and evaluation for each case study.

4 CASE STUDY I: RELIABILITY OF MONITORING PLATFORMS

Monitoring is essential in modern network management to identify problems in underlying communication infrastructures of modern organizations. However, current monitoring systems are unable to recover their internal faulty entities forcing the network administrator to manually fix the occasionally broken monitoring solution. This case study, therefore, address this issue by introducing a self-healing monitoring solution. The proposed solution combines the availability and communication transparency provided by P2P-based overlays with self-healing properties.

The definition of self-healing property investigated in this case study is based on the description given by Berns and Ghosh (BERNS; GHOSH, 2009), where self-healing is focused in maintaining or restoring a system's safety property. In addition, those authors defined that self-healing systems are the ones that guarantee healing from a limited subset of all actions that can affect the global behavior of the system. Then, in this case study, the self-healing property is focused on both maintaining and restoring the monitoring capacity of a management system when crash failures occurs on the monitoring services or on the management entities hosting those services.

The solution here presented is described considering a scenario of a monitoring system for a Network Access Control (NAC) installation. The next sections depict the self-healing P2P-based approach, its instantiation to constitute a NAC monitoring installation, the evaluation of the solution, and the analysis of the proposed approach in the light of the compliance to the management requirements pursued in this thesis, the achievement of the integration issues, potentialities, and shortcomings.

4.1 Self-Healing P2P-Based Approach

The objective of the self-healing P2P-based approach is to provide reliability for management platforms, in especial network and services monitoring ones. Reliability, here, is understood as the ability of maintaining the proper operation of the platform. The proper operation comprises two actions: the identification of broken management entities (*e.g.*, monitors, agents) and the recovering process of the functionalities executed by those faulty entities (*e.g.*, monitoring, event correlation).

The approach proposed to enable the identification and recovering processes is twofold. First, it embeds self-healing properties inside the peers, and, in a complementary fashion, it uses the connectivity capabilities of the peers to support the healing process. This complementary behavior can be enabled if the following assumptions are ensured.

- The underlying infrastructure of the management platform comprises a P2P overlay, turning the final management platform into a P2P management overlay.
- The management tasks are exposed as services inside the P2P management overlay.
- The self-management of the management tasks must be transparent for the tasks themselves.
- The P2P management overlay must provide mechanisms for service discovery, and for dealing with peer group interactions.

Using those assumptions as the high level directives of the approach, it is possible to define the concepts and architecture that enables the fusion of self-healing properties and P2P infrastructures to design a reliable network and service monitoring platform. However, before describing the self-healing P2P solution itself, the next section describes and limits the type of failures that can be healed by the here solution proposed.

4.1.1 Supported Types of Failures

Taking into account the definition given by Berns and Ghosh (BERNS; GHOSH, 2009) for the term self-healing, there must be a limited set of actions (in this case failures) from what the system itself is able to autonomously heal. There are different types of failures, and according to Tanenbaum and Van Steen (TANENBAUM; STEEN, 2007) there are five failure models, that are described below.

Crash failure. This is the type of failure where an entity (*e.g*, server, process) stops working suddenly but it was working properly until it stopped. An important feature of crash failures is the fact that once the entity stopped, nothing is heard from it anymore.

Omission failure. In this case, the entity fails to respond to incoming request. An omission failure can be divided into: receive omission, where the entity fails to receive an incoming message; or send omission, where the entity fails to send messages.

Timing failure. This kind of failure occurs when a response message lies outside a specific real-time interval. Two examples of timing failures are: when the response arrives too soon which forces the existence of buffers to store the responses; and when the response arrives too late.

Response failure. This is the type of failure where the response is simply incorrect. There are two kinds of response failures: one where the value of the response is wrong, and the other is known as state transition failure where an entity deviates from the correct control flow to a wrong one.

Arbitrary or Byzantine Failure. This is the most difficult type of failure to be treated because the entities suffering from this failure start to behave in an unpredictable manner. For example, servers could start producing outputs that would never happen, or servers could start working in a malicious manner.

The self-healing P2P approach proposed in this case study is limited to support the failure models described in Table 4.1. The choice to support **crash failures**, **omission**

Table 4.1: Type of failures supported by the self-healing P2P approach

Type of Failure	Type of Support
Crash failure	Full support
Omission failure	Full support
Timing failure	Partially supported
Response failure	Not supported
Arbitrary or Byzantine Failure	Not supported

failures, and to partially support **timing failures** was based on the analysis of the characteristics of the failure models and the integration issues of self-healing P2P approach.

For example, peers that host the monitoring services (*i.e.*, the management tasks) of the P2P management overlay could leave such overlay at any time. This situation can be understood as a crash failure of the monitoring service that was running inside such peer and the management overlay should heal the crashed monitoring services. Now, if the message that verifies the health of the monitoring services is lost this does not mean that a particular monitoring service (supposed to send or receive such message) is in failure. Thus, omission failure is an important failure model to be supported by the self-healing P2P solution. To finalize, timing failure is supported in the case where a message arrives too soon, but is not addressed in the case when the message arrives too late. The self-healing P2P approach is able to buffer early messages, but is not able to rollback an ongoing healing process when a late message arrives from the entity considered to be in failure. The next sections describe the mechanisms used to support the above listed failure models and the self-healing P2P approach itself.

4.1.2 Architecture and Concepts

The self-healing architecture is based on a P2P management overlay formed on top of the monitored devices and services. The use of P2P functionalities provides a transparent mechanism to enable communication targeted to publish, discover, and access management tasks inside the overlay. The control of such communications is delegated to the P2P framework used to implement the architecture. The P2P infrastructure helps on distributing the identification of failures and also to provide scalability on the recovery process.

The main architectural elements (management services, self-healing and self-configuration services) and the monitoring environment are illustrated in Figure 4.1. In addition, there are two concepts that are important for the design of the self-healing P2P-based approach: instances of management services and management peer group. In fact, management services and these two concepts were defined by Panisson *et al.* (PANISSON et al., 2006) and they are revised in favor of the self-healing P2P-based approach as described below.

Management Services. Name given to the management tasks to be executed by the management peers of the P2P overlay. Examples of types of management services are: monitoring network services (*e.g.*, email servers, DNS[1]), configuration of firewalls, monitoring operational status routers (*e.g.*, interfaces, queues, congestion).

Instances of Management Services. For a certain type of management service, there might exist more than one peer executing the tasks associated to this management

[1]Domain Name Service.

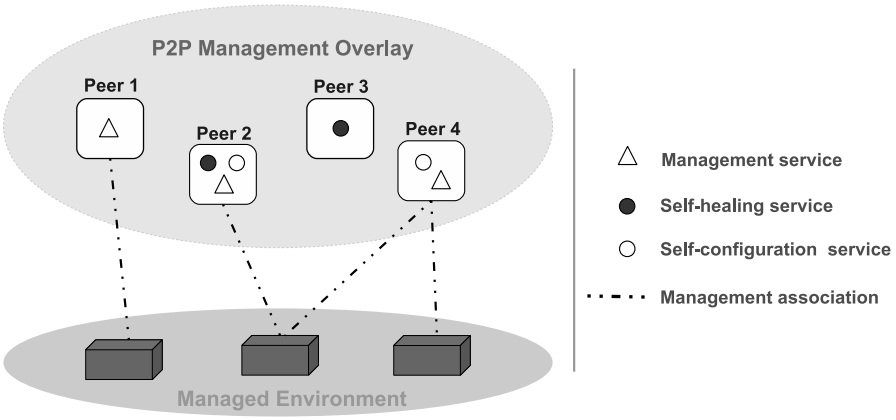


Figure 4.1: Self-healing P2P-based environment and architecture

service. Thus, the P2P management overlay can support the execution of instances of management services. Those instances will be executing the same code, and can interact either with other instances of the same management task or with instance of different types of management tasks. For example, Figure 4.1 shows that the management service represented by the triangle has 3 instances, while self-healing and self-configuration services have 2 instances each one.

Management Peer Group (MPG). Instances of the same management task form a management peer group. The number of instances of management services is transparent for the entities that are willing to use/invoke the services/tasks offered by the management service. In this sense, the management group concept acts as a shield for the instances of the management service, and just the operations of this service are exposed for the rest of the P2P management overlay. The details of the architecture and load balancing inside the management peer group are described by Panisson *et al.* (PANISSON et al., 2006). Considering the environment presented in Figure 4.1, the instances of the management service (represented by the triangle) form, automatically, a management peer group inside the P2P management overlay. In the same way, the instances of self-healing and self-configuration services form their own management peer groups.

Self-healing service. This is a management service designed to start the recovery process of instances of management services that were detected to be broken. In general terms, this service can be seen as an ordinary management service, subjected to the same rules and behavior of any other management service inside the P2P management overlay. However, instead of managing network devices and services, the self-healing service manages the management services of the P2P overlay, by regulating the necessity of activating or not a recovery process.

Self-configuration service. Designed to find available management peers, inside the P2P management overlay, to instantiate the management service detected to be broken. Together, the self-healing and self-configuration services provide the recovering process of the self-healing property.

The management service is able to heal itself if, after the crashing of some of its instances (possibly due to a peer crash), new instances become available, thus recovering

the service and guaranteeing its availability. In order to cope with that, two functions must be supported: failure detection and service instance activation. To ensure that the failure detection and instance activation functions work properly, two requirements must be filled on the P2P management overlay.

- First, each management service (including the self-healing and self-configuration services) must run at least 2 instances to enable the detection and recovery in case of problems on the management service. That is so because a single faulty instance cannot react itself if it is crashed, then at least another instance is required.
- Second, each peer must not host more than one instance of the same management service in order to avoid several instances of that service crashing if the hosting peer crashes too.

The maintenance of the management infrastructure is assured while those requirements are fulfilled.

4.1.3 Failure Detection

Failures in a management service are detected by a self-monitoring procedure embedded inside each management service. Each service instance, in intervals of t seconds, sends a signal (heartbeat) to all other instances of the same management service (inside the management peer group that they form) to announce it is running. Self-monitoring, in this sense, means that there is no external entity monitoring the instances of a management service deployed inside the overlay. Indeed, the instances of the management service themselves can monitor their liveness throughout the heartbeat messages. So, if one instance crashes, the other instances will miss the former's heartbeats and then will initiate the process to recover this instance. Figure 4.2 illustrates what happens inside each management peer group when some failure happens.

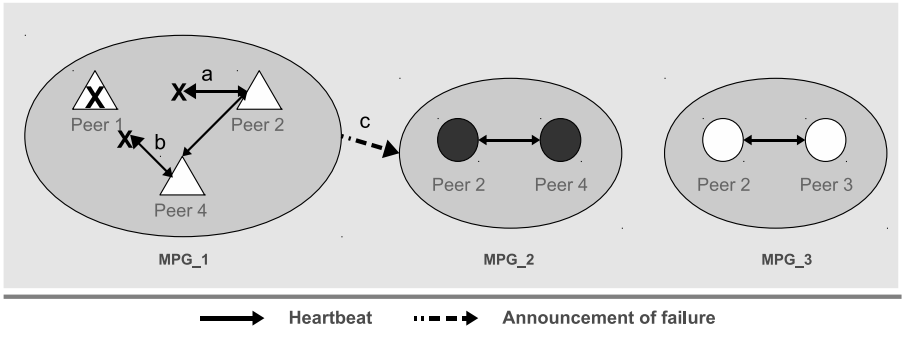


Figure 4.2: Failure detection inside management peer groups

The environment depicted in Figure 4.2 is a management peer group representation of the P2P management overlay presented in Figure 4.1. There are 3 management peer groups: MPG_1 (Management service), MPG_2 (Self-healing service), and MPG_3 (Self-configuration service). Periodically, the instances of each service sends and receives the heartbeats. However, whenever a failure happens, like illustrated by the message "a" and "b" in Figure 4.2, the other instances start the process to determine if one instance of the management service is really missing and needs to be recovered. The failure detection procedure is presented in the Algorithm 4.1.

Algorithm 4.1 Failure detection algorithm

Require: $PG = \{p_1, p_2, ..., p_i\}$ set of peers that form a peer group
Require: $MS = \{ms_1, ms_2, ..., ms_j\}$ set of management services
Require: $SPG = \{PG_1, PG_2, ..., PG_k\}$ set of peer groups
Require: $MPG = \{mpg_1, mpg_2, ..., mpg_l\}$ where $mpg_l = \{ms_j, PG_k\}$, $ms_j \in MS$ and $PG_k \in SPG$
Ensure: $i > 0$ where i is the index of the PG
Ensure: $j > 0$ where j is the index of the MS
Ensure: $k > 0$ where k is the index of the SPG
Ensure: $l > 0$ where l is the index of the MPG
Ensure: $t > 0$ where t is the interval to send heartbeats
Ensure: $s > 0$ where s is the interval to listen heartbeats of other instances
Ensure: $r > 0$ where r the interval to wait for the answer of a suspicious failed peer

```
1:  loop
2:    WaitHeartbeatCycle(t)
3:    for all mpg_l ∈ MPG do
4:      num_heartbeats ← 0
5:      retry_contact ← false
6:      num_mpg_instances ← GetNumberPeersInsideMPG(mpg_l)
7:      SendHeartbeatToMPG(mpg_l)
8:      ListenHeartbeats(num_heartbeats, mpg_l, s)
9:      if num_heartbeats < (num_mpg_instances − 1) then
10:       suspicious_peer ← p_i ∈ PG_k from mpg_l without answer
11:       DeclareSuspiciousFailure(suspicious_peer)
12:       RetryContactWith SuspiciousFailedPeer(retry_contact, r, suspicious_peer)
13:       if retry_contact = false then
14:         failed_management_service ← ms_j from mpg_l
15:         DeclareFailedInstance(suspicious_peer, mpg_l)
16:         InformSelfHealingService(failed_management_service, mpg_l)
17:       end if
18:     end if
19:   end for
20: end loop
```

Heartbeats that get lost in the network may wrongly suggest the unavailability of a service instance. Instead of immediately assuming an instance as down given the lack of a heartbeat, it first becomes suspect by the other instances (lines 10-11 of Algorithm 4.1). In order to double check the availability of the suspicious instance, one of the other alive instance tries to contact the suspicious instance back[2] (line 12 of Algorithm 4.1). If no contact is possible, the suspicious instance is finally declared unavailable (lines 13-17 of Algorithm 4.1), and the announcement of such failure is sent to the self-healing management peer group[3] (as illustrated by message "c" in Figure 4.2).

Assuming s as the time spent to receive the heartbeats, and r the time spent to double

[2]It is out of the scope of this thesis investigate the mechanisms inside the management peer group to define which of the alive peers will be elected to execute the retry process. An example of how management peer group can be designed to support such kind of actions is described by Panisson (PANISSON, 2007).

[3]The details related to the P2P overlay communications presented in the lines 6-8, 11, 12, 15, 16 of Algorithm 4.1 are described in Appendix A.

check the availability of a suspicious instance, the maximum detection time is $dt = t+s+r$. The distribution of heartbeats from one service instance to all others is accomplished using group communications. At the network level, in the best case, group communication is supported by multicast communications. In this case, the number of heartbeat messages h issued by i service instances in t seconds will be $h = i$. However, if multicasting is not available, the notifying service instance is forced to send, via unicast, copies of the same heartbeat to all other instances. In this case, the number of messages will be $h = i^2 - i$. In this way, the presence of multicast directly influences the network traffic generated by the failure detection function.

Failure detection is essentially a consensus problem (BENEDIKTSSON; SWAIN, 1992) (OLFATI-SABER; FAX; MURRAY, 2007) (AYSAL; BARNER, 2009). Solutions on this topic, coming from the dependability field, could be employed to model and validate the detection approach (IZUMI; SAITOH; MASUZAWA, 2004) (KAR; MOURA, 2009) (DING et al., 2009). Instead of that, this case study employs a practical approach of actually implementing the aforementioned heartbeat schema.

4.1.4 Service Instance Activation and Policies

Instance activation is crucial to recover the management service that lost some of its instances. It is on instance activation that the self-healing and self-configuration services, presented in Figure 4.1, play a key role. Once an instance detects a remote crashed one, it notifies the *self-healing service* (Algorithms 4.2 and 4.3) that determines how many, if any, new instances of the faulty service must be activated. To do so, the self-healing service internally checks a repository of service policies that describes, for each management service, the minimum number of instances that must be running, as well as the number of new instances that must be activated once the minimum boundary is crossed (lines 2-4 from Algorithm 4.3). An example of a repository of service policies is illustrated in Table 4.2. The P2P management overlay considered in this example, is composed of 3 management services.

Table 4.2: Service policy repository

Management service	Minimum instances	Activate instances
Management service A	2	1
Management service B	2	2
Management service C	2	1

As listed in Table 4.2, the management service “A” must have at least 2 instances running. In case of failure, one new instance must be activated. Analyzing the case of the management service “B”, on the other hand, although 2 instances are running, whenever activation is required 2 other new instances will be initiated. If the number of remaining running instances of a service is still above the minimum boundary, the self-healing service ignores the faulty service notifications. For example, in the case that management service “C” is the one represented in Figure 4.1, if a single instance crashes no action will be executed because the remaining 2 instances do not cross the minimum boundary. It is assumed that policies are defined by the system administrator and transferred to the self-healing service instances long before any failure occurred in the P2P management overlay. Some detailed explanation about the maintenance of policies in a P2P management overlay are discussed by Nobre and Granville (NOBRE; GRANVILLE, 2009).

Algorithm 4.2 Self-healing - Main control loop

Ensure: $failed\_mngt\_service$ stores the identification of the management service notified to be in failure

1: **loop**
2: $failed\_mngt\_service \leftarrow \emptyset$
3: WaitForFailureNotification($failed\_mngt\_service$)
4: PolicyEvaluationActivation($failed\_mngt\_service$)
5: **end loop**

Algorithm 4.3 Self-healing - Policy evaluation and activation

Require: $POLICY = (min\_instances, num\_activations)$
Require: $POLICYSET = \{POLICY_1, POLICY_2, ..., POLICY_i\}$ set of policies
Require: $MS = \{ms_1, ms_2, ..., ms_j\}$ set of management services
Require: $MSPOLICY = \{msp_1, msp_2, ..., msp_k\}$ where $msp_k = \{ms_j, POLICY_i\}$, $ms_j \in MS$ and $POLICY_i \in POLICYSET$
Ensure: $i > 0$ where i is the index of the $POLICYSET$
Ensure: $j > 0$ where j is the index of the MS
Ensure: $k > 0$ where k is the index of the $MSPOLICY$
Ensure: $failed\_mngt\_service$ stores the identification of the management service notified to be in failure

1: **loop**
2: $num\_instance \leftarrow$ GetNumberOfInstances($failed\_mngt\_service$)
3: $(min\_instances, num\_activation) \leftarrow (min\_instances, num\_activation) = POLICY_i$, for $POLICY_i \in msp_k$ where $ms_j = failed\_mngt\_service$
4: **if** $num\_instance < min\_instances$ **then**
5: **for** $i = 0$ to $num\_activation$ **do**
6: CallSelfConfiguration($failed\_mngt\_service$)
7: **end for**
8: WaitSelfConfigurationAnswer()
9: **end if**
10: **end loop**

Algorithm 4.4 Self-configuration - Recovery process

Ensure: $failed\_mngt\_service$ stores the identification of the management service notified to be in failure
Ensure: $num\_retries$ stores the number of attempts to find an available peer
1: $attempts \leftarrow 0$
2: $deployed \leftarrow false$
3: **while** $(attempts < num\_retries) \vee (deployed = false)$ **do**
4: **if** FindAvailablePeer($failed\_mngt\_service$) $= true$ **then**
5: DeployInstance($failed\_mngt\_service$)
6: $deployed \leftarrow true$
7: **else**
8: $attempts \leftarrow attempts + 1$
9: **end if**
10: **end while**
11: **if** $deployed = true$ **then**
12: **return** $true$
13: **else**
14: **return** $false$
15: **end if**

Once required, the self-healing service tries to activate the new instances defined in the service policy (lines 5-7 of Algorithm 4.3) by contacting the *self-configuration service*. Such configuration service is then responsible for creating new instances of the faulty service on peers that do not have those instances[4], as depicted by Algorithm 4.4.

A peer hosting solely a self-configuration service can be seen as an spare peer ready to active new instances of any service in failure. Thus, different than the failure detection function, instance activation is performed outside the management peer group that contains the failing management service. That is so because decoupling the instance activation function allows more flexibility to deal with the number of components for each function, and this directly impacts the number of message exchanged in the overlay.

4.2 Development of the Case Study

This case study addresses the problem of monitoring systems that lack self-healing feature by considering a Network Access Control (NAC) (LóPEZ et al., 2007) installation. This section characterizes the NAC environment, the management platform called ManP2P used to provide the underlying infrastructure for the NAC monitoring system, and the implementation of the self-healing P2P-based approach inside ManP2P platform.

4.2.1 NAC Monitoring System

A NAC installation is composed of devices and services (*e.g.*, routers, firewalls, RADIUS[5] servers) that control how users and devices join the institutional network. A typical NAC environment and its associated monitoring system is presented in Figure 4.3.

Traditional monitoring systems (*i.e.*, without self-healing support) fail to protect NAC,

[4]The details related to the P2P overlay communications presented in the line 2 of Algorithm 4.3 and line 4 of Algorithm 4.4 are described in Appendix A.
[5]Remote Authentication Dial In User Service.

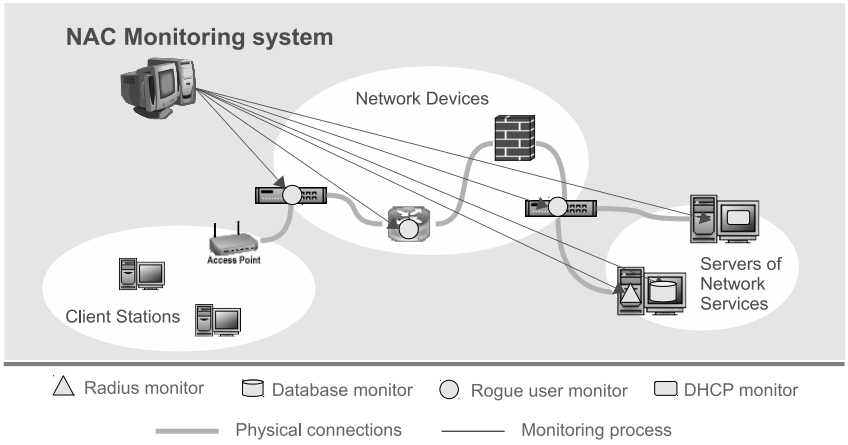


Figure 4.3: NAC environment

for example, in two situations. First, consider a crashed RADIUS server whose associated RADIUS monitor crashed too. In this case, the administrator reacts to the RADIUS problem only when users complain about unsuccessful login attempts. Worse than that, however, is the second situation. Suppose a failure in the *rogue user* service, responsible for detecting unregistered devices, and another failure in the monitor associated to it. In this case, unregistered devices will silently join the network without generating user's complains. In contrast to the first situation, the "silent failure" remains because no signal is issued either by network users or, and most seriously, by the broken monitoring system.

The employment of the self-healing P2P-based approach is able to address both situations discussed above. Before describing the implementation of such approach, it is necessary to introduce the main characteristics of the P2P management overlay employed to provide the underlaying infrastructure. The monitoring system of a NAC installation is deployed on the ManP2P platform (PANISSON et al., 2006). The details of such platform and the deployment of the NAC monitoring system are discussed as follows.

4.2.2 ManP2P Platform

The management overlay ManP2P is a previous work, and its architecture has been described by Panisson *et al.* (PANISSON et al., 2006). Due to the fact that the ManP2P architecture is based on management services and peer group interactions, it is used, here, to provide the underlying infrastructure required for the development of the self-healing P2P-based approach. In fact, in this case study, the ManP2P functionalities are extended in order to explicitly support self-healing processes.

The collection of management peers forms the ManP2P management overlay. Each peer runs basic functions (*e.g.*, granting access to other peers or detecting peers that left the P2P network) to maintain the overlay structure. In addition, each peer hosts a set o management services instances that execute management tasks over the managed network (in the specific case, monitoring tasks). A management service is available if at least one single instance of it is running on the overlay. More instances of the same service, however, must be instantiated in order to implement fault tolerance. Figure 4.4 exemplifies a scenario where management services (LDAP[6] monitors, Web servers monitors, rogue user monitors) for a NAC installation are deployed on the ManP2P management overlay.

[6]Lightweight Directory Access Protocol.

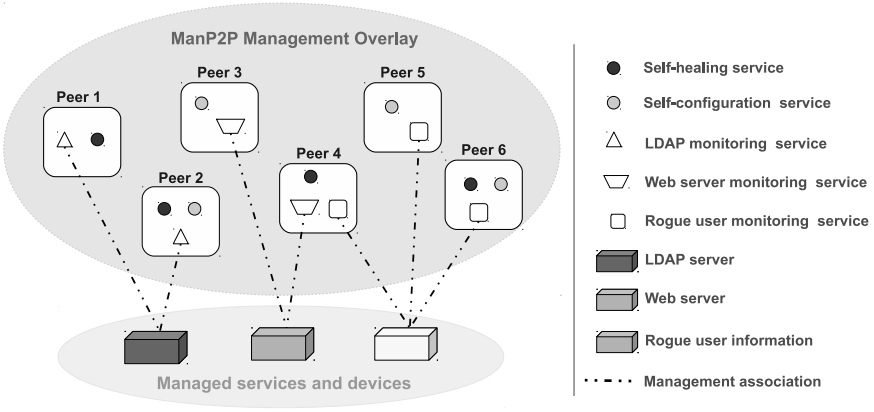


Figure 4.4: ManP2P environment for a NAC monitoring installation

In Figure 4.4, peers #1 and #2 host service instances, pictured as a triangle, that monitor an LDAP server. Peer #4, on its turn, contacts both the Web server and the rogue user service because it hosts management services to monitor these elements. Each peer, in summary, may host different services at one. In the extreme cases, there could exist peers with no management services (thus useless peers) or peers hosting one instance of each available management service (this possibly becoming an overloaded peer).

4.2.3 Implementation

The implementation of the architecture, in an actual monitoring system, is based on the previous code of ManP2P. Figure 4.5 depicts the internal components of a peer on ManP2P platform with the introduced support for the self-healing solution.

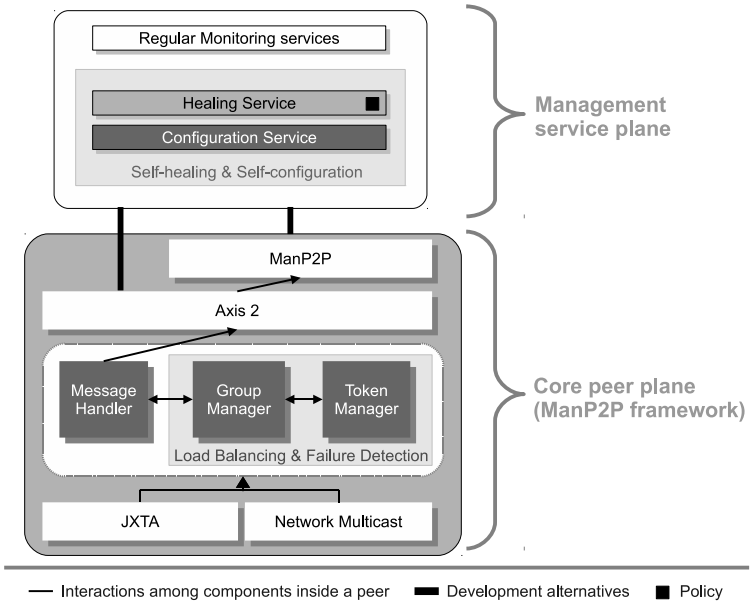


Figure 4.5: Peer architecture to support self-healing

Components are divided into the *core peer plane* and *management service plane*. The core peer plane's components are responsible for controlling the communication mecha-

nisms between peers. At the bottom the *JXTA* and *network multicast* components implement group communication using unicast (via JXTA) or network multicast. On top of them, the *group manager* and *token manager* components control, respectively, group membership and load balancing (via a virtual token ring). Messages are handled by the *message handler* component that interfaces with Axis2 (FOUNDATION, 2010) to communicate with the management service plane's components. A ManP2P component on top of the core peer plane is used to implement complementary functionalities.

At the management service plane the regular monitoring services are found. Although located in this plane, monitoring services themselves do not monitor remote instances for fault detection; this verification is in fact performed by the group manager component. That is so because the self-monitoring function is designed to be native in any peer, freeing the developer of new management services to concentrate their efforts on the management functionalities he/she is coding without worrying about the self-monitoring support. At the management service plane the self-healing and self-configuration services are also found. As mentioned before, they are responsible for activating new instances of monitoring services when required. The black little square inside the self-healing service represents the policies that define the minimum number of instances of each management service, as well as the number of new instances that must be activated. Peers and internal monitoring services have been coded in Java using Axis2, JXTA, and ManP2P previously developed libraries. Monitoring services have been specifically developed as dynamic libraries that can be instantiated when required by a hosting peer.

4.3 Experimental Evaluation

The evaluation is performed in terms of recovery time when fail-stop crashes occur in the monitoring system. In addition, the traffic generated by the communication between the elements of the solution is also measured. This leads to the contribution of showing the trade-off between the recovery time and associated network traffic. The determination of such trade off is important because it shows when a faster recovery process consumes too much network bandwidth. On the other side, it also shows when excessively saved bandwidth leads to services that remain unavailable longer.

The effects of the failures are evaluated considering variations on: (a) the number of simultaneously crashing peers, (b) the number of peers in the management overlay, and (c) the number of management services running on the overlay. The experiments were executed in a high performance cluster, called LabTec from the GPPD research group at UFRGS (GPPD, 2008), from which 16 nodes were used to host the management peers. The recovery time and the generated traffic have been measured capturing the P2P traffic and timestamping it using a packet capture `tcpdump` software. Traffic volume is calculated considering the headers and payload of all packets generated by the system operations. Recovery time has been measured 30 times for each experimental case and computed with a confidence interval of 95%.

Although the size of P2P systems is typically of scales much higher than 16 nodes, it is assumed that in an actual management scenario of a single corporation, administrators tend not to use a large number of managing nodes. In this way, 16 peers are sufficient for most actual management environments. Over the P2P management overlay 12 different NAC management services were deployed (namely, monitors for LDAP, DNS, DHCP, Radius, data base, Web servers, rogue user, firewall, proxy, access point, switches, and routers). In addition, the self-healing and self-configuration special services were also

instantiated. The single service policy enforced in all management services of the experiments defines that at least 2 instances per service must be running and, in case of failures, just another one must be activated per crashed instance. Considering the above, two main sets of experiments have been carried out: multiple crashing peers, and variable number of peers and services. The same measurement process was adopted for both experiments. The details of this process, and the description of the sets of experiments and their associated results are described as follows.

4.3.1 Measurement Process

The major challenge of the measuring process is the distributed nature of the experiments. The nodes of the cluster employed on the experiments were attached to a switch. The traffic of each node, in this way, was isolated and this situation prevents the use of a single device capturing the traffic of the entire nodes for further analysis[7]. Another option would be to trust in a global synchronized clock for all the nodes of the cluster. However, the initial experiments showed that this option was not appropriated too.

The solution designed for measuring the traffic consumption and the detection and recovery times was based on a log service inside a dedicated node of the cluster. Every time an event happens inside the experimental scenarios, then a remote invocation for logging this event was executed. Figure 4.6 illustrates the sequence of events and measurement points that are logged whenever a peer crashes and the self-healing mechanism starts.

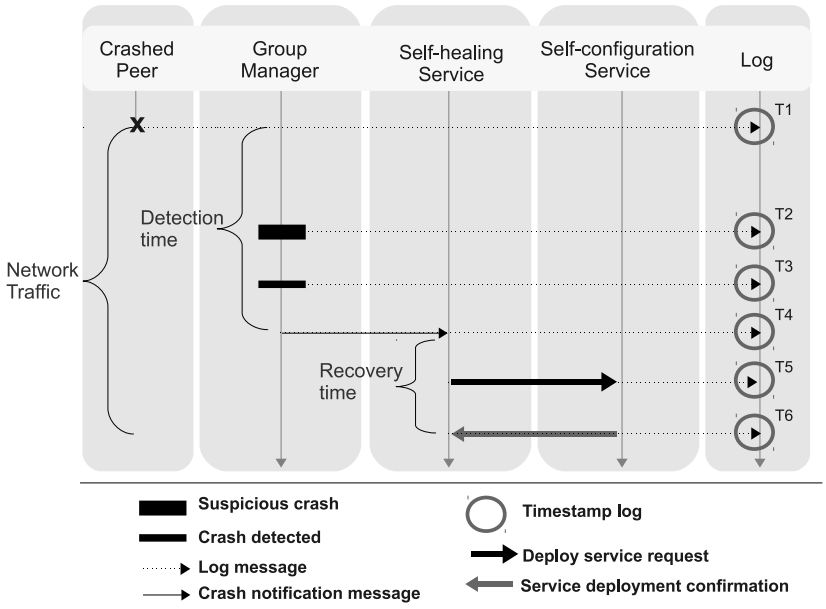


Figure 4.6: Measurement process

A crash, in the experiments, is provoked by a script that kills the peer instance running inside a cluster node. In addition, right after killing the peer instance, this script signals (by sending a ping message) all the nodes involved in the experiments and logs the failure of a peer. This signal is used later on the analysis to determine the start point of a recovery process (represented by “T1” in Figure 4.6).

[7]Mirroring the traffic from all nodes to a single port of the switch was not possible due to limitations of such device.

The first group management component of a peer belonging to the same management peer group of failed instance of a management service will signal the log service about a suspicious crash (“T2”). After electing the instance inside the management peer group that will execute the failure detection actions (“T3”), the group management component of the peer associated with the elected instance confirms the crash on the log service and activates the the self-healing service (“T4”). The measurement process is finished when the self-configuration service signals the log (dashed line), the self-healing service (solid line) illustrated in “T6” of Figure 4.6), and activates an script to signal all nodes about the end of one cycle of recovering process (by sending another ping message).

The time results presented in the graphics of the sets of experiments are retrieved from the analysis of the information inside the log service, and they represent the interval between “T1” and “T6”. The traffic results are based on the analysis of the transmitted packets of the cluster nodes composing the P2P management overlay between the interval of the two pings that signaled the start and stop of the recovery process. In the sequence, each one of the set of experiments and their associated results are presented.

4.3.2 Multiple Crashing Peers

The first experiment was designed to check the performance of the self-healing monitoring architecture when the number of simultaneously crashing peers hosting management services increases until the limit where half of them are broken. In addition, it is checked whether the number of instances of the self-healing and self-configuration services influences the recovery time and generated traffic.

For this set of experiments, the following setup was used: 12 management services are always deployed, each one with 2 instances running on the overlay. The total 24 service instances (*i.e.*, 12×2) are placed along 8 peers, each one thus hosting 3 (*i.e.*, $24 \div 8$) service instances. The number of crashing peers varies from 1 to 4. Since each peer hosts 3 instances, the number of crashing instances varies from 3 (12.5%) to 12 (50%), out of the total of 24 instances. Additional 4 peers have been used to host the self-healing and configuration services. Their varying number of instances has been organized, in pairs of self-healing/configuration, as follows: 2 and 4 instances, and 4 and 4 instances. Finally, it is considered that group communication support is implemented interchangeably using multicast and unicast.

Figure 4.7 shows, in seconds, the time taken by the monitoring system to detect and activate new instances of the crashing services using the “spare” cluster nodes that host the self-configuration service. The first occurrence of 3 crashing services correspond to the situation where 1 peer fails; 6 crashing services correspond to 2 failing peers, and so on. No value is provided in 0 (zero) because with no failing peers there will not be any recovering service.

The recovery time as a function of the number of crashing peers stayed mostly constant. With that, it can be concluded that the system scales well considering a management scenario of 16 nodes. There is a little variance on the recovery time as a function of the self-healing and self-configuration services. In fact, such difference is the result of employing multicast or unicast. When peers use multicasting they quickly become aware of changes in the system, and can rather react faster. Using unicast, however, more messages are sent, delaying the communication and, as a consequence, the reactions. In summary, the recovery time is not strongly influenced either by the self-healing and self-configuration services or by the number of crashing services. There is, however, a little influence from the use of multicast or unicast in the group communication support.

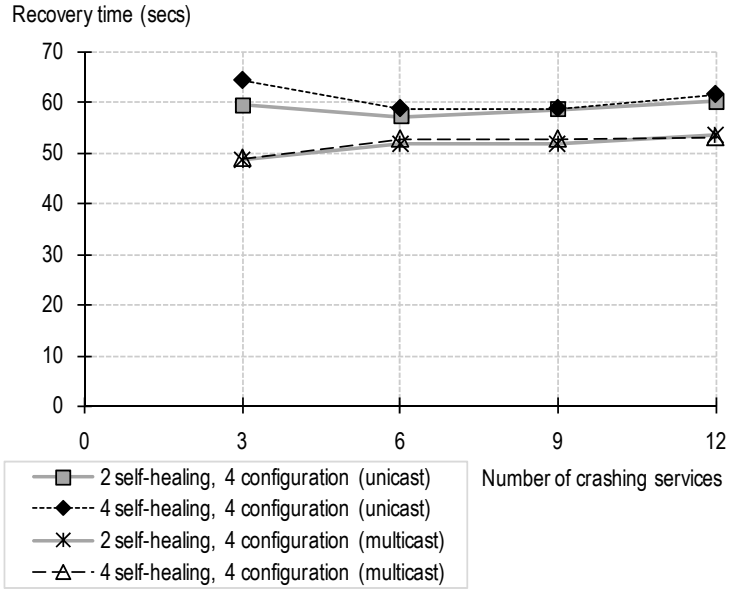


Figure 4.7: Recovery time with multiple crashing

Figure 4.8, in its turn, presents the network traffic generated by the management overlay in the recovery process. In this case, for 0 (zero) there exists an associated network traffic because, in the self-monitoring process, heartbeat messages are constantly sent regardless the presence or not of a failure.

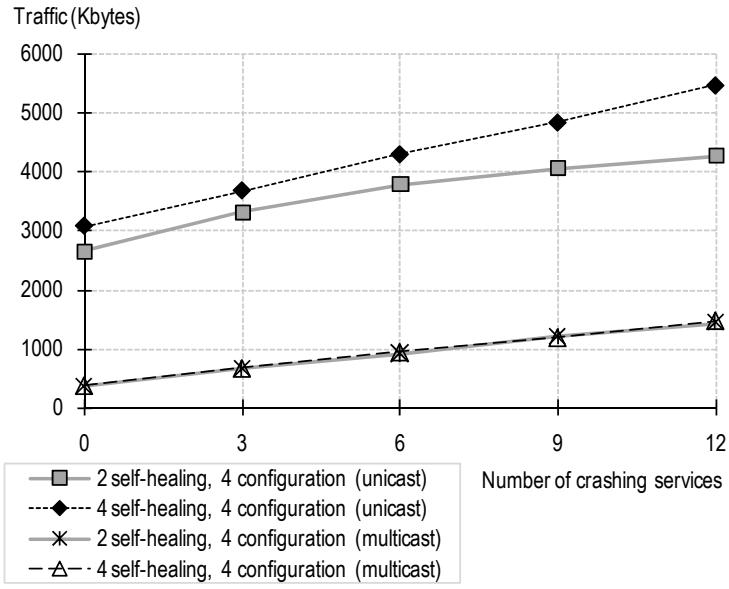


Figure 4.8: Traffic to recover crashing peers

Network traffic, in its turn, presents a stronger influence of multicast or unicast support. As can be observed in Figure 4.8, multicast-based communications saves more bandwidth, which is expected. The important point to be observed, however, is that with the increasing number of crashed services the traffic generated to recover them is closely linear, and even when doubling the number of failures, the traffic generated does not dou-

ble together. Although not so efficient as in the case of recovery time, the bandwidth consumption is still scalable in this case.

Putting these two parameters together (*i.e.*, recovery time and bandwidth consumption) and observing the graphs, if multicasting is used the number of self-healing and self-configuration services and the number of crashing peers do not influence the recovery time, and slightly increase the bandwidth consumption. In the case of unicast, however, the option of employing 2 self-healing instances instead of 4 is better, because this setup reacts slightly faster yet generating less traffic.

4.3.3 Varying Number of Peers and Services

The second experiment shows the relationship between recovery time and generated traffic when single crashes occur (which tends to be more frequent than multiple crashes), and the number of peers and services varies. It is considered the recovery process when the number of management services increases from 1 to 12 (*i.e.*, from 2 to 24 instances) over three setups where 2, 6, and 12 peers are used to host the management services. In addition to single crashes, it is also fixed the number of 2 self-healing and 2 self-configuration services instances, hosted by 2 peers. This is done because, as observed before, the number of such instances few impacts on the recovery time.

In Figure 4.9, where the recovery delay is presented, services communicating via multicast are depicted with dashed lines, while services using unicast are depicted with solid gray lines. The recovery time, when only 2 peers are employed, is usually higher because each of the 2 peers hosts more service instances. When one of the peers crashes, more instances need to be activated. On the other extreme, with 12 peers, each peer hosts less services, leading to the situation where a crashing peer actually triggers the activation of less service instances. The fact that more instances need to be activated, as the result of a more loaded peer, can be observed in Figure 4.10, that shows the traffic generated to recover the system.

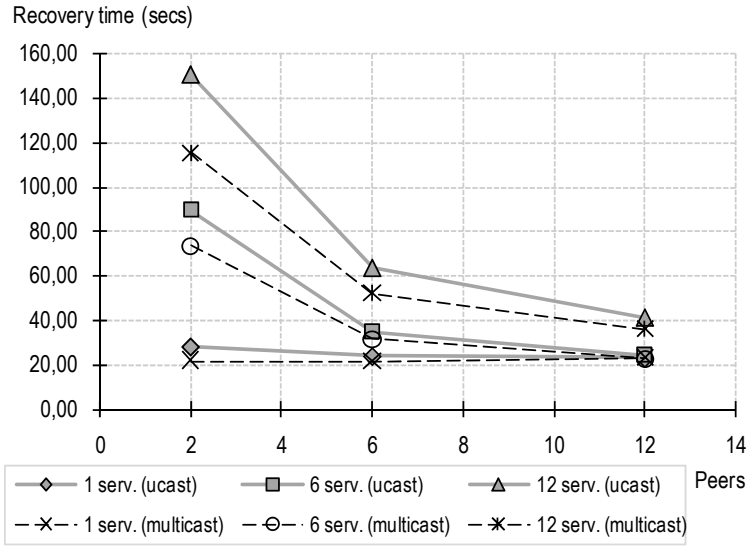


Figure 4.9: Recovery time for multiple peers

Again, multicast communications save more bandwidth than unicast, as expected. However, it is important to notice that now the number of services in each peer influ-

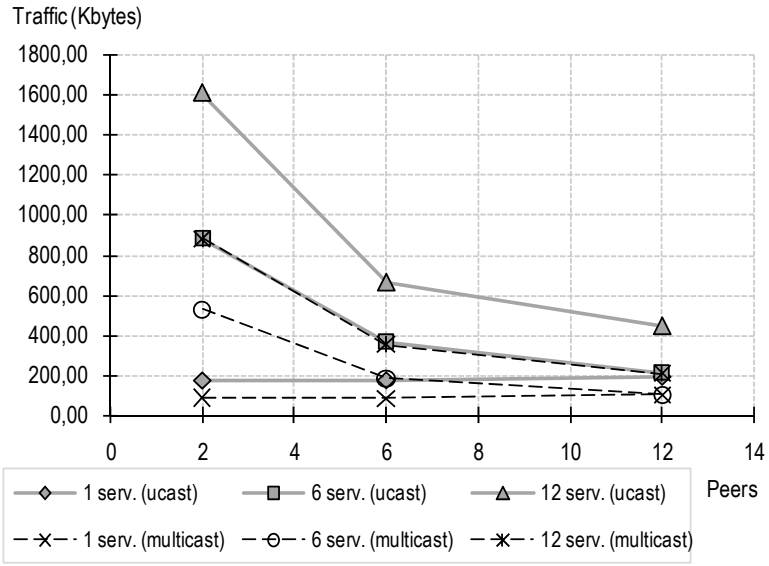


Figure 4.10: Recovery traffic with multiple peers

ences too. For example, 6 instances running on the same peer, via multicast, (line “6 serv. multicast", with 2 peers in the x axis) still takes longer and generates more traffic to recover the system than the case where, via unicast, only 1 service is deployed (line “1 serv. unicast", with 2 peers in the x axis). This confirms that the number of peers and service instances must be similar in order to recover more promptly the system without generating too much traffic. If an administrator is restricted in terms of peers available, he/she must try to restrict the number of services employed as well. If new services are required, however, the option of also increasing the number of peers should be considered.

Now considering the whole picture, administrators should worry neither about simultaneous crashes nor about the number of self-healing and self-configuration services. Increased multiple crashes are more scare, and even if they happen the system is able to recover reasonably fast. As observed, the number of self-healing and configuration services does not affect the overall performance of the system. However, administrator should do pay attention to the number of available peers and service instances, as mentioned before. Finally, the employment of multicast and unicast in the group communication mechanism influences the recovery time (less) and the generated traffic (more). Choosing multicast whenever possible helps to improve the response time of the system. Unfortunately, multicast is not always available, which forces the administrator to use unicast to implement group communication.

4.4 Critical Evaluation of the Designed Approach

The objective of this section is to present an analysis of the self-P2P approach designed for this case study in terms of: the compliance to the network management requirements established in Section 3.2, the degree of accomplishment of the issues related to the integration of self-* properties and P2P, and the potentialities and shortcomings associated with such approach.

4.4.1 Compliance to Management Requirements

In this thesis, there are four management requirements that lead to the employment of a self-* P2P solution, and thus, they must be provided by the designed solutions. Therefore, this section is devoted to analyze the self-healing P2P solution developed in this case study in the light of the accomplishment of the network management requirements considered in this thesis. A summary of the analysis is present in Table 4.3, while the detailed explanation is depicted in the following paragraphs.

The first management requirement is related to the efficient use of the resources (that is understood as network traffic or processing power). In this matter, the experiments of the self-healing P2P solution proposed for keeping the proper execution of monitoring platforms showed that the traffic load inserted by the developed solution follows a linear curve on the network traffic consumption. An example of such efficiency in term of traffic consumption is observed when the number of services to be healed doubles, but the self-healing P2P solution does not require the double of the network traffic to heal the services. In addition, the experiments showed that there is a trade-off between the recovery time and the consumption of network traffic. This means that the proposed solution enables a fine configuration of the "prices" that the administrator wishes to pay for fast or slow healing of his/her monitoring platform. This way, the meaning of "efficient use of resources" can be configured by each administrator, instead of being fixed by the solution.

Table 4.3: Achievement of management requirements on self-healing P2P case study

Requirements		**Accomplishment**
Efficient Resource Usage		Linear consumption of resources with the increase of simultaneous healing processes.
		Present a trade-off between fast healing against traffic consumption.
Agile Management Actions		No human intervention is required to identify failures and recover the monitoring overlay.
Transparency	Access	-
	Location	After healing the failed services, their new location is transparent for the MPG.
	Migration	Failed services are moved to different peers without affecting the access to those services.
	Relocation	While an instance of a failed service is recovered/relocated the other instances still provide the resources associated to the service.
	Replication	-
	Concurrency	-
	Failure	Whenever there are at least two instances of the same service running in distinct peers, then it is possible to hide the failure of an instance.
Parallel and Simultaneous Behavior		Healing occurs in different parts of the overlay at any time and place and they are triggered following an horizontal parallelization.

Agile management actions is the second management requirement, as presented in Table 4.3. The solution proposed in this case study removes from the human administrator

the task of verifying the proper execution of the management platform. The self-healing P2P approach is able to identify a failure on a service of such platform and to heal the service inside another available peer without the intervention of the administrator. The consequence is the agile reestablishment of the functionalities of the platform.

The third management requirement is associated with the transparency of the management actions in the perspective of the users of the network or service being managed. There are different types of transparency and Table 4.3 describes the ones important to be achieved in the context of this case study. For this reason, access, replication, and concurrency types of transparency are not part of the self-healing P2P solution, while location, migration, relocation, and failure are the important types of transparency to be provided by the solution here proposed[8]. In fact, there is a strong relationship among the transparency types provided by the self-* P2P solution of this case study. There are a set of facts that contribute for this relationship: (i) a P2P overlay is the infrastructure of the management platform; (ii) a management peer group (formed by at least 2 instances of the same service) is the cell to execute the services of the management platform; (iii) the existence of the management peer group is transparent to the user requesting the service of such group. Then, based on this facts it is possible to provide the four transparency types detailed in table 4.3.

Parallel and simultaneous behavior is the last management requirement here considered. In this case study, this behavior is achieved due to the decentralization of the mechanisms that identify failures and that heal the system. As described in previous sections, the identification of failures is designed to be embedded inside the communication mechanism of each management peer group. Thus, the identification of failures becomes an horizontal process inside each management peer group, without the existence of any external entity. Yet, the embedded design inside the management peer groups enables different groups to identify failures and start the healing process simultaneously with the support of the self-healing and self-configuration services. This way, the parallel and simultaneous behavior is native to each peer of the management overlay.

Summarizing, this section presented that the compliance of the management requirement was achieved by the self-healing P2P solution developed in this case study. Thus, next step on the critical evaluation of the designed approach is to analyze the degrees of achievement of the integration issues. The following section presents such analysis.

4.4.2 Achievement of Integration Issues

Section 3.4 presents the issues that have to be chased in order to develop a network management solution integrating self-* properties and P2P. The self-healing P2P solution designed for this case study pursued those issues. Table 4.4 presents the comparison of the self-healing P2P design with the characteristics of the integration issues. This comparison is presented in terms of accomplishment degrees of an integration issue and for each attributed degree there is a justification.

The employment of local information was the only issue that could not be accomplished on the self-healing P2P design. The explanation for the negative result is associated with the nature of the self-healing property. When a peer or a management service instance fails, it is not possible anymore to use its local information. Thus, external and distributed information is required so that the self-healing element can perform its task.

On the other and, the success of achieving the design of light self-elements is attributed

[8]In this case study, access, replication, and concurrency types of transparency are provided by the implementation of the ManP2P platform, but they are out of the scope of the proposed self-healing P2P solution.

Table 4.4: Degree of integration issues for self-healing P2P approach

Integration Issue	Accomplishment Degree	Justification
Common knowledge of management task	Partially achieved	Self-elements are homogeneous in the management peer group perspective, but there are two heterogeneous self-elements.
		Knowledge split among group management component of peers and the instances of self-healing and self-configuration services.
Local information	Not achieved	The information that starts the self-healing process depends on messages arriving from remote instances of the failed management services.
Parallel and distributed algorithms for decision-making	Partially Achieved	Solely distributed algorithms are used for the detection and recovery process.
		The decisions are taken by single instances of the self-healing and self-configuration services.
Light self-elements	Achieved	The logic of the self-healing and self-configuration services is very simple and their functions are specialized.
Reduction of explicit and global coordination	Partially Achieved	Explicit messages starting the recovering process are required.
		Existence of intra and inter group coordination, but no global coordination.

to the decision of decoupling the processes of evaluating whether a recover is required, from the recovering process itself. The specialization of each one of these processes, in different elements, enables the reduction of group communication on the management infrastructure running the self-healing P2P approach.

In essence, the recovery capacity is given by the self-configuration service, that is responsible for finding an available peer where a failed instance of a management service can be deployed. An available peer means a peer executing the self-configuration service but not executing an instance of the failed service. In this sense, the more self-configuration instances are deployed, the higher are the chances to find an available peer. However, the bigger is the management peer group, the more heartbeat messages will be exchanged among the instances inside the group. Thus, coupling the tasks of self-configuration and self-healing in the same self-element could bring more overhead, in

terms of traffic consumption, for the management infrastructure. Therefore, the self-healing P2P approach, here designed, decoupled the evaluation of an activation of a new instance from the task of activating this instance. This can give flexibility to configure the self-healing infrastructure according to the desires of the network administrator.

However, the simplicity of self-elements requires, as expected, more efforts on the design of interactions among those elements in order to accomplish the management task. In this way, the maximum degree accomplished by light self-elements negatively impacts the results in the following integration issues: common knowledge of management task, parallel and distributed algorithms for decision-making, and reduce the explicit and global coordination. The main reasons of this negative impact are: there are no homogeneous management tasks, single instances drive the decisions, and intra and inter group communications are required.

4.4.3 Potentialities and Shortcomings

Despite the advantages of the self-healing P2P solution, discussed in the previous sections, there are extra potentialities that can be explored. One of the first potentialities of the self-healing P2P solution is its capacity for easily expanding the management infrastructure in terms of number of management services inside the P2P overlay. This can be achieved using the service policies to reconfigure the management platform with minimum human intervention. The original role of the service policy is to be verified whenever a failure notification arrives at the self-healing service. Now, to increase the number of instances of management services, it is possible to define a management service for the management of the policies that can interact with the self-healing to start the instantiation process of new instances according to the changes on the policies.

Moreover, it is possible to define a self-destructing service that is responsible for shrinking the management infrastructure. In this case, instead of activation of services there will be deactivation of services. One possible motivation for shrinking the system could be saving energy. For example, if during the night the percentage of failures is very low, it is not worth to use large number of monitors, self-healing, and self-configuration instances, and thus, those instances can be hosted by a few number of peers, letting the extra peers in standby or turned off.

The self-destruction service could also help the self-healing P2P approach to heal and restore the proper operation of the management overlay after the occurrence of group partitioning problems. In this case, if the peers composing the P2P management overlay are for some reason partitioned into two groups, and if inside each one of those groups there are instances of the self-healing and self-configuration services, then each group will start the healing process of the monitoring services of the management overlay. Then, when the problem that caused the partitioning is solved and the two groups are again part of the same management overlay, there will be the double of the necessary monitoring services. Those extra services could be disabled/halted by the self-destructing service after the partitioning problem was solved.

Nevertheless, there are security problems associated to the capability of expanding and shrinking a management infrastructure. Indeed, the self-healing P2P approach, as described in this case study, suffers from the lack of security strategies. For example, there is no security mechanism to avoid that a malicious peer starts to send messages directly to the self-configuration service asking for the deployment of instances of management services that, in fact, did not failed. This malicious peer could lead to an scarcity of resources (*i.e.*, non available peers), so that, when a legitimate failure needs to be recovered, there will

be no peers available to be used. An authentication and access rights mechanism is required in the self-healing P2P design to define which kind of service is allowed to execute specific tasks, like deployment of new instances, or killing new instances.

Although the self-healing P2P approach is not prepared to react against malicious behavior against the self-configuration service, there are malicious behaviors against the self-healing service that can be treated by the self-healing P2P approach. For example, monitoring services could start sending arbitrary request to the self-healing service asking for the deployment of new instances of monitoring services, that in fact did not failed. This behavior can be described as a Byzantine failure, and this kind of failure is identified by the self-healing service at the moment it evaluates if it is really necessary to deploy a new instance. The evaluation process of the self-healing service helps on identifying that a Byzantine failure is happening, but currently there is no mechanism to heal the P2P management overlay from this kind of failure.

Another potentiality associated with the self-healing P2P approach is the re-usability of the self-configuration service. In a first moment, the objective of designing this service was to help on the deployment of failed instances of management services. However, there is no restriction on the self-healing P2P solution, that prevents the use of this service to deploy instances of management services that need to be available for the first time inside the P2P management overlay. In this way, it is possible to think about self-organizing or self-adapting services that use the self-configuration service to help, respectively, on the organization and distribution of the management services inside the management overlay, or on replacing different types of such services to cope with changes on the management environment. Once again, security is the major shortcoming of expanding the functionalities of the self-configuration service.

4.5 Final Remarks

This chapter presented the design and evaluation of a self-healing P2P approach that was employed for building up a NAC monitoring system. The solution achieves self-healing capacity by splitting in two different processes the functions of failure detection and system recovery. Failure detection is executed inside management services that monitor final devices, while system recovery relies on special services called self-healing (that decides when new service instances must be activated) and self-configuration (that activates the new service instance as a reaction for the self-healing service decision).

Based on the results of the experimental evaluations, it is possible to conclude that the number of instances of the self-healing and self-configuration services is not a major player in the performance of the system. These results also allows to state that simultaneous crashes on the management services does not influence so expressively the system performance either. A network administrator willing to employ a self-healing monitoring solution should not concentrate his/her efforts in finding an ideal number of self-healing and self-configuration services. The experiments employed 2 and 4 instances, respectively, and the system response was satisfactory. The fact that must be observed, however, is the group communication solution available on the managed network: multicast turns recovery faster while consuming less network bandwidth. Unfortunately, IP multicast can not always be provided, and unicast ends up being chosen in such situations.

The most important aspects that must be observed in a self-healing P2P solution are the number of peers employed in the P2P management overlay and the number of service instances deployed. With few instances, there is no need for several peers. On the other

hand, with a large number of instances the number of peers should grow consistently, otherwise, on the occurrence of a failure, the recovery time will be higher and more network bandwidth is consumed by the intensive P2P traffic generated.

In addition to the remarks about the performance of the developed self-healing P2P solution, it is also important to highlight the cooperative design embedded in such solution. Taking into account the definitions associated with cooperation in Section 3.3, the self-healing P2P solution presents the intentional cooperative posture, and to achieve this posture it uses the following methods of cooperation: grouping, communication, specialization, collaborating by sharing information, and coordination of tasks.

The intentional posture is justified by the fact that the management peer groups are embedded with the common action of identifying failures and looking for the special services, *i.e.*, self-healing and self-configuration. Thus, to provide this intentional posture, it was necessary to create groups (with at least 2 instances for each management service) that can communicate, share information, and finally coordinate tasks with the specialized services to actually provide the feature of healing failed services inside a management platform. In essence, without the cooperative behavior of the services inside the peers there is no self-healing property on the system, *i.e.*, an isolated element on the environment can not provide alone this property. This way, only through the cooperation among the elements of the environment, the self-healing property can emerge on the management platform.

5 CASE STUDY II: RESOURCE MANAGEMENT OF NETWORK VIRTUALIZATION

Network virtualization is an emerging trend claimed to reduce the costs of future networks. The key strategy in network virtualization is of slicing physical resources (links, routers, servers, etc.) to create virtual networks composed of subsets of these slices. One important challenge on network virtualization is the resource management of the physical or substrate networks. Sophisticated management techniques should be used to accomplish such management. The sophisticated techniques offered by self-management approaches rise as an appropriated alternative to address the challenges of managing the efficient use of substrate resources on network virtualization.

This chapter, therefore, depicts the joint use of self-organizing property and P2P technique to manage the substrate network resources. First, the concepts behind self-organizing P2P approach are presented in a generic manner, *i.e.*, not strictly devoted to a network virtualization environment. In the sequence, the network virtualization model considered in this thesis is introduced, and then the instantiation of the generic self-organizing P2P solution on the network virtualization environment is depicted. The implementation and evaluation of the solution are described. Finally, a critical evaluation of the self-organizing P2P solution is discussed and the final remarks of this case study presented.

5.1 Self-Organizing P2P Approach

Rather than using solely the connectivity of P2P infrastructures and keeping the network management approach based on hierarchies and delegation (as exposed in Section 2.3), the self-organizing P2P approach, here proposed, strongly explores the design of cooperative P2P interactions to accomplish the self-organizing tasks. The design of such approach is grounded on the identification of manageable elements (*e.g.*, network traffic, management applications) with complementary perspectives. For example, there is a peer "A" where a traffic flow *t* is generated, and there is a peer "B" receiving this traffic. In this case, the manageable element traffic *t* has complementary perspectives, because for peer "A" it is an outgoing traffic and for peer "B" it is an incoming traffic. In this case, there is a relationship between the outgoing and the incoming perspectives of traffic *t* that can be cooperatively found by those peers.

The basic condition that enables the employment of self-organizing P2P approach in an environment is the possibility of identifying situations where peers "A" and "B" can locally analyze the manageable element features and then cooperate to find a match associating this element to both peers, and this match can lead to reorganization actions. The

manageable element considered in this thesis is the network traffic of a given environment, and the objective of the self-organizing P2P solution is to reorganize the placement of the entities related to such traffic (*e.g.*, source or destination), so that load balancing and even traffic reduction on the environment can be achieved. The self-organizing P2P solution is based on the assumptions listed below.

- There are peers generating network traffic and peers receiving such traffic.
- Either the source or the destination of the traffic (or even both) can be moved towards each other.
- The network traffic associated to a peer can be identified as source, destination, and cut-through (traffic passing through the peer).
- The initial deployment of a network environment is not addressed by this work. It is assumed that a different, external planning tool analyzed the conditions of physical resources and then choose the best initial placement for the elements of the network environment.
- The network topology defined by the first placement will not change during the lifetime of the network environment, even after the reorganization of the peers.
- The reallocation associated with the entities related to the manageable elements must be as much transparent as possible for the final users.

Based on these assumptions, and inspired on self-organization techniques presented in (HERMANN, 2007) the self-organizing P2P approach is defined. This approach is composed of two control loops: monitoring and self-organizing. The *monitoring control loop* is responsible for gathering local information that is used on the analysis and determination of which is the behavior assumed by a certain traffic flow. The type of the monitored information and its gathering point depends on (i) the type of network environment where the self-organizing P2P approach is being employed and (ii) the type of manageable element. In this thesis, the monitored information, associated to the manageable element, is the number of bytes transmitted and received in the network layer and associated to each flow in the transport layer (considering a TCP/IP architecture).

The *self-organizing control loop* is the core component, where is executed the self-organizing algorithm. Such algorithm is responsible for analyzing the complementary perspectives, and, whenever a match is identified, it starts the re-organization of the elements of the network. In this thesis, two complementary perspectives are associated to the network traffic flows. *Receiving candidate* is the perspective where a peer has to receive the resources (*e.g.*, application) associated to a certain traffic flow that needs to be moved. *Moving candidate* is the complementary perspective that regards the identification of a traffic flow that should be eliminated from the peer where it was identified. Considering the high level concepts of the self-organizing P2P approach, a moving candidate can be either the source or the destination of the traffic. The decision of which specific role (*i.e.*, source or destination) will be moved depends on the nature of the environment employing the proposed approach.

Five stages characterize the self-organizing control loop, as illustrated in Figure 5.1. On the first stage, the capacity of the links associated to a peer are locally analyzed under both perspectives of the self-organizing algorithm. The analysis starts with the identification of overloaded links. Then, the traffic associated to each one of these overloaded links

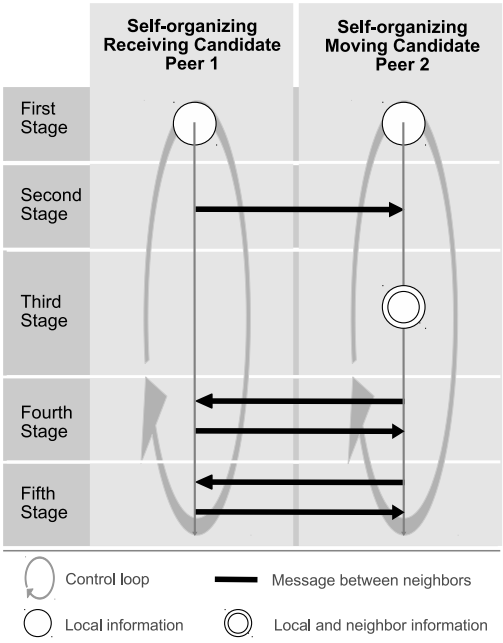


Figure 5.1: Self-organizing algorithm

is investigated on the light of receiving and moving candidate perspectives[1]. So far, the output of the first stage is a list of traffic flows that should be moved or received by the peers executing the self-organizing algorithm.

On the second stage, if the list of receiving and moving candidates is not empty, the peer executing the receiving candidate perspective (Peer 1 in Figure 5.1) adopts an active behavior while the other perspective (Peer 2 in Figure 5.1) adopts a passive behavior. The receiving candidate perspective sends a message to its neighbor requesting to receive a traffic flow it supposes that is deployed on such neighbor. Meanwhile, the peer executing the moving candidate perspective waits for a message from a neighbor requesting to receive a traffic flow belonging to its moving candidate list.

The third stage is characterized by the decision of moving an application associated to the traffic flow. Whenever the peer executing the moving candidate perspective receives the request message, it verifies if there is a match between the traffic flow requested and its internal list of flows to be moved. Whether there is a successful match, the moving candidate perspective calculates the costs of re-organizing the elements. If there is a favorable cost-effect relationship on this re-organization, the request to move is accepted. The forth stage is the announcement of the decision to move and the reservation of resources to host the resources associated to the traffic flow at the peer that had its request accepted. Finally, at the fifth stage, the reorganization is executed.

This generic description of the self-organizing P2P approach presents the key concepts of the cooperation between the peers to find a match to re-organize the placement of elements inside the network environment. It is remarkable that not all kinds of network environments can cope with this approach, specially the ones where the placement of network elements follows very strict requirements. Nevertheless, next section presents a suitable environment for the employment of the self-organizing P2P approach.

[1]Heuristics have to be designed to identify the perspectives to be analyzed for each time the self-organizing P2P approach is instantiated.

5.2 Network Virtualization Model

According to recent researches, virtualization is a promising technique to deploy future networks (CHOWDHURY; BOUTABA, 2009) (NIEBERT et al., 2008)(FEAMSTER; GAO; REXFORD, 2007)(BERL et al., 2008). Its key idea is the identification and separation of two roles: a substrate provider, who owns and maintains the substrate network, and a virtual provider, who builds its own infrastructure by renting slices of resources from the substrate provider. Looking at a virtual provider as an entity selling services, the advantage of virtualization relies on the fact that costs in running a physical infrastructure can be outsourced to an external provider.

One important point is the definition of the main characteristics of an architecture for virtualization. It should be noted that this thesis presents the minimal assumptions of such architecture, and further details can be found in specific projects like GENI (ELLIOTT; FALK, 2009) or 4WARD (4WARD, 2010). A key aspect in the architecture of virtual networks is the transparency: virtual nodes cannot see or exchange any type of information, in order to assure isolation of the networks of different providers. Additionally, the data exchanged in the virtual network is transparent to the substrate provider to preserve the privacy of the customers. Nevertheless, some minimal primitives to inspect the activity of the different slices are normally available: as an example, primitives to allow the controller of the substrate resources to know the actual usage of computational resources and traffic consumption. Figure 5.2 shows the architecture of a substrate node, where resources are sliced and assigned to different virtual providers.

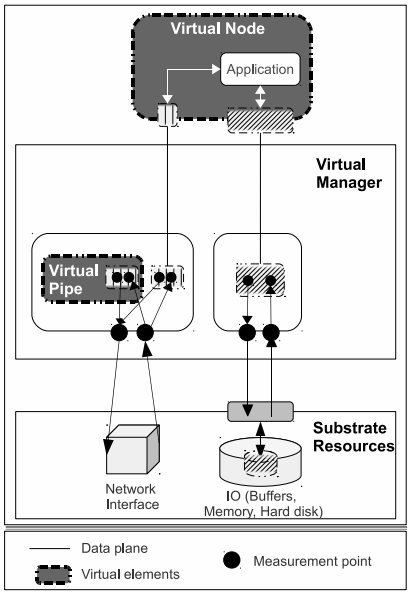


Figure 5.2: Substrate node architecture

The network virtualization architecture, considered in this thesis, is composed of three modules: substrate resources, virtual manager, and virtual elements (virtual nodes and virtual pipes). Inside a substrate node, there must exist the substrate resources and virtual manager modules. On the other hand, the existence of virtual modules is flexible, in the sense that it is possible that a substrate node hosts solely virtual nodes, or virtual pipes, or it can host both virtual elements from different virtual providers. It is assumed that it must not host both virtual node and pipe from the same virtual provider (the explanation

for this assumption is described later on the text). Figure 5.2 illustrates the details of the modules and components of the substrate node architecture.

The substrate resources module comprises the physical resources of the substrate node, like network interfaces, IO (Input/Output) system (*e.g.*, buffers, memory, hard disk), and CPU. The virtual manager behaves as a middleware between the physical resources and the resources that are attributed to the virtual elements. As observed in Figure 5.2, virtual elements are the virtual nodes and the virtual pipes. There is a conceptual difference between these virtual elements related to what is physically deployed at the substrate network and what is perceived by the virtual network. An example that helps on clarifying this difference is presented in Figure 5.3, where two virtual network topologies and the architecture of the substrate nodes supporting these virtual topologies are illustrated.

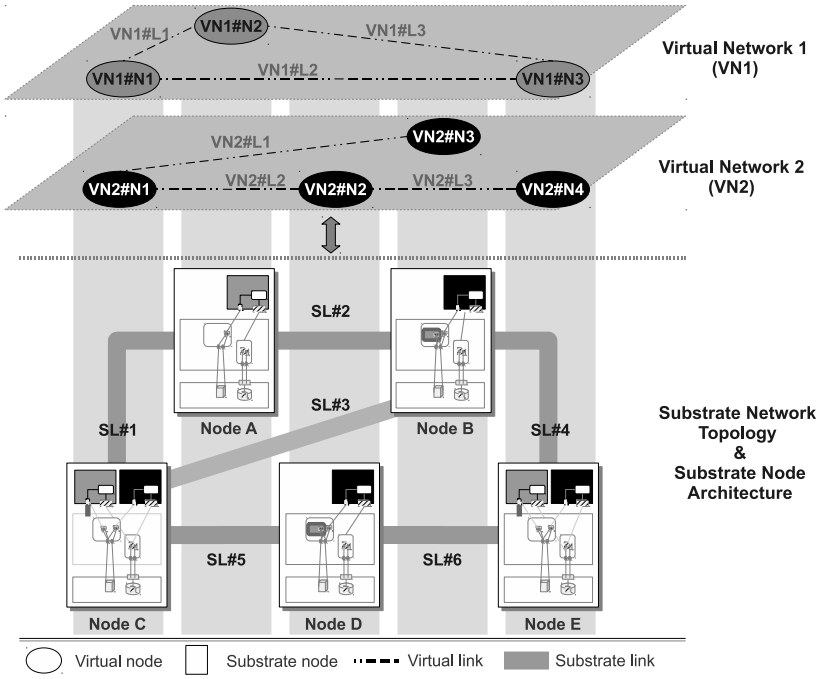


Figure 5.3: Substrate and virtual networks perspectives

The substrate network in Figure 5.3 is composed of five substrate nodes (“A”, “B”, “C”, “D”, “E”), and six substrate links (“SL#1”, “SL#2”, “SL#3”, “SL#4”, “SL#5”, “SL#6”). The Virtual Network 1 (VN1) is composed of three virtual nodes, “VN1#N1”, “VN1#N2”, and “VN1#N3”, being each one respectively mapped to the substrate nodes “C”, “A”, and “E”. The second virtual network, Virtual Network 2 (VN2), is composed of 4 virtual nodes, “VN2#N1”, “VN2#N2”, “VN2#N3”, and “VN2#N4”, respectively mapped to the substrate nodes “C”, “D”, “B”, and “E”.

A virtual node corresponds to a set of physical resources from the substrate node that are wrapped together and assigned to a virtual network. Another virtual element is also deployed in the architecture of the substrate nodes “B” and “D”, but it is not illustrated in VN1 topology. This element is the virtual pipe. While the virtual node is designed to belong to both virtual network topology and substrate node architecture, the virtual pipe is designed to be part solely of the substrate node architecture and transparent for the virtual network topology. Indeed, a virtual pipe is a “connection” between two non physically adjacent virtual nodes.

As observed in Figure 5.3, not always two virtual nodes are allocated in adjacent substrate nodes. For example, the virtual nodes of VN2 are all logically and physically adjacent, *i.e.*, logically due to the virtual topology, and physically because they are deployed in adjacent substrate nodes. However, the tuples of virtual nodes <VN1#N1,VN1#N3> and <VN2#N1,VN1#N3> of VN1 are only logically adjacent, since the virtual links "VN1#L2" and "VN1#L3" connecting the virtual neighbors are mapped to more than one substrate links. For example, "VN1#L2" is actually mapped to substrate links "SL#5" and "SL#6", while "VN1#L3" comprises substrate links "SL#2" and "SL#4".

There is an economical reason that encourages the use of both virtual pipes and virtual nodes. The deployment of a virtual node requires the assignment of more substrate resources than the deployment of a virtual pipe, because virtual pipes require resources solely for "tunneling" traffic and no other complex infrastructure (*e.g.*, applications). The assignment of substrate resources means that costs will be charged to enable the use of these resources. Thus, the higher is the number of virtual nodes, the higher are the total costs to deploy a virtual network. Moreover, depending on the geographical demands, it is not cost-effective to deploy two physically adjacent virtual nodes, but somehow the traffic of one virtual node must arrive in its logically adjacent neighbor. In this sense, virtual pipes represent a less expensive connection between logically adjacent virtual nodes.

Inside a virtual pipe runs essentially cut-through traffic, *i.e.*, a traffic that is not originated inside the substrate node itself, but it just passes through the network interfaces of the substrate node. This means that for each virtual pipe associated to the traffic of a virtual network there is at least two substrate links that are being used but actually could be spared if the logically adjacent virtual nodes were also physically adjacent. Thus, this situation justify the employment of the self-organizing P2P approach to reduce the amount of cut-through traffic inside the substrate network by moving virtual nodes.

5.3 Development of the Case Study

The components of the self-organizing P2P approach, *i.e.*, monitoring and self-organizing control loops, are instantiate inside the virtual manager module, as showed in Figure 5.4.

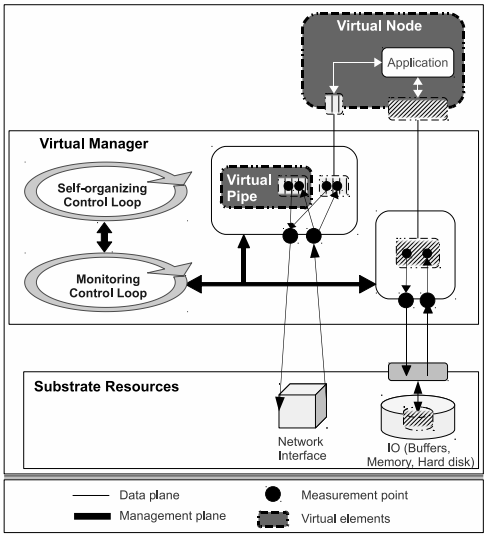


Figure 5.4: Self-organizing and the substrate node architecture

The self-organizing P2P solution is based on a parallel and distributed algorithm that uses local information during the decision-making. This algorithm is executed by the self-organizing control loop inside the virtual manager module of each substrate node (which constitute a peer), as illustrated in Figure 5.4. The local information is retrieved from the measurement points by the monitoring control loop. Afterwards, the self-organizing and the monitoring control loops exchange the measured information to verify whether a re-organization of virtual elements is required. Such re-organization is triggered by the detection of an overloaded substrate link and the identification of a cut-through traffic.

Two complementary perspectives can characterize a cut-through traffic: the virtual pipe where the traffic is passing by and the virtual node that is the source generating this traffic. Analysis of traffic under the virtual pipe perspective is called *receiving candidate perspective*, while analysis considering virtual nodes is called *moving candidate perspective*. Considering the fact that inside a substrate node might exist virtual nodes and virtual pipes, it becomes necessary to execute both perspectives inside the same control loop. In this way, the cut-through traffic associated either to virtual nodes or virtual pipes can be properly identified and self-organizing actions activated. The instantiation of the self-organizing algorithm, and heuristics to identify the receiving and moving candidates are presented in the next sections.

5.3.1 Self-organizing Control Loop

The self-organizing control loop instantiated for network virtualization environment is detailed in Algorithms 5.1, 5.2 and 5.3. The first stage of the self-organizing algorithm is the characterization of the conditions that trigger the reallocation of the virtual resources, and it comprises the lines 3-32 of Algorithm 5.1. An overloaded link is the element that starts the evaluation of a possible re-organization of virtual resources on the substrate network. So, each substrate link sl_i of a substrate node is analyzed according to the condition established at line 5 of Algorithm 5.1. This condition says that if the total traffic of sl_i ($T(sl_i)$) is greater than the threshold (calculated with $alpha$) established based on the channel bandwidth ($B(sl_i)$), then this substrate link sl_i is considered overloaded and is added to the set overloaded substrate links $OvSubsLink$.

The next step is to identify which virtual link inside the overloaded substrate link is using the major amount of bandwidth. This step is described by the lines 9-16 of Algorithm 5.1, and the output is the set of virtual links vl_j overloading the substrate link sl_j ($OvVirtualLink\_sl_i$). In the sequence, starts the process to determine whether the flow associated to the virtual link matches a cut-through traffic pattern (lines 19-32 of Algorithm 5.1). To determine this kind of match, two heuristics are employed and will be detailed later on. The lists $Receiving\_Candidate\_List$ and $Moving\_Candidate\_List$ are the outputs of the matching process, and based on those lists, the next stages of the self-organizing algorithm are described in the Algorithms 5.2 and 5.3.

The virtual manager executing the Algorithm 5.2 first determines which are the virtual networks inside the $Receiving\_Candidate\_List$ (line 1 of Algorithm 5.2), then it execute the operations to request a virtual node to the substrate neighbor linked to the most overloaded virtual link vl_j^k of each virtual network (lines 2-7 of Algorithm 5.2). It is possible that more than one virtual link of the same virtual network were identified as overloaded. In this case, the virtual manager has to choose solely one virtual link associated to the virtual network under analysis.

Algorithm 5.1 Self-organizing - Main control loop

Require: $SL = \{sl_1, sl_2, ..., sl_i\}$ set of substrate links of a substrate node
Require: $sl = \{vl_1, vl_2, ..., vl_j\}$ set of virtual links inside a substrate link
Ensure: $OvSubsLink$ is the set of overloaded substrate links
Ensure: $OvVirtualLink\_sl_i$ is the set of overloaded virtual links from an sl_i
Ensure: $T(link)$ function that returns the total traffic of a given $link$
Ensure: $B(link)$ function that returns the bandwidth capacity of a given $link$

1: **loop**
2: WaitSelfOrganizingCycle()
3: $OvSubsLink \leftarrow \emptyset$
4: **for all** $sl_i \in SL$ **do**
5: **if** $T(sl_i) > B(sl_i) * \alpha$ **then**
6: $sl_i \in OvSubsLink$
7: **end if**
8: **end for**
9: **for all** $sl_i \in OvSubsLink$ **do**
10: $OvVirtualLink\_sl_i \leftarrow \emptyset$
11: **for** $vl_j \in sl_i$ **do**
12: **if** $T(vl_j) \geq T(sl_i) - T(vl_j)$ **then**
13: $vl_j \in OvVirtualLink\_sl_i$
14: **end if**
15: **end for**
16: **end for**
17: $Receiving\_Candidate\_List \leftarrow \emptyset$
18: **for all** $OvVirtualLink\_sl_i$ **do**
19: **for all** $vl_j \in OvVirtualLink\_sl_i$ **do**
20: **if** ApplyReceivingCandidateHeuristic(vl_j) **then**
21: $vl_j \in Receiving\_Candidate\_List$
22: **end if**
23: **end for**
24: **end for**
25: $Moving\_Candidate\_List \leftarrow \emptyset$
26: **for all** $OvVirtualLink\_sl_i$ **do**
27: **for all** $vl_j \in OvVirtualLink\_sl_i$ **do**
28: **if** ApplyMovingCandidateHeuristic(vl_j) **then**
29: $vl_j \in Moving\_Candidate\_List$
30: **end if**
31: **end for**
32: **end for**
33: {Execute in parallel the next stages of the self-organizing algorithm}
34: **if** $Receiving\_Candidate\_List$ not $\emptyset$ **then**
35: Launch Receiving Candidate Algorithm
36: **end if**
37: **if** $Moving\_Candidate\_List$ not $\emptyset$ **then**
38: Launch Moving Candidate Algorithm
39: **end if**
40: **end loop**

Algorithm 5.2 Receiving candidate algorithm

Require: $VNET = \{vnet_1, vnet_2, ..., vnet_k\}$ is set of virtual networks
Ensure: vl_j^k is a virtual link j of the $vnet_k$
Ensure: $neighbor$ is the identification of a substrate node
Ensure: $match[]$ is an array that stores the answers of the requests for moving the virtual node of $vnet_k$
Ensure: t is the timeout to wait for an answer of a neighbor

```
1: VNET ← IdentifyVNETs(Receiving_Candidate_List)
2: for all vnet_k ∈ VNET do
3:    vl_j^k ← MostOverloadedVirtualLink(vnet_k)
4:    neighbor ← GetSubstrateNeighborIdentification(vl_j^k)
5:    match[k] ← false
6:    RequestVirtualNodeAndWait(neighbor, vnet_k, match[k], t)
7: end for
8: for index = 0 to k do
9:    if match[index] = true then
10:      PrepareMigration(vnet_index)
11:      GetVirtualNode(neighbor, vnet_index)
12:      SendConfirmationOfVirtualNodeMigration(neighbor, vnet_index)
13:   end if
14: end for
```

Algorithm 5.3 Moving candidate algorithm

Require: $VNET = \{vnet_1, vnet_2, ..., vnet_k\}$ is set of virtual networks
Ensure: $request\_arrival[]$ is an array to store the arrived requests to move a virtual node of $vnet_k$
Ensure: t is the timeout to wait for an answer of the neighbor
Ensure: $neighbor$ is the identification of a substrate node

```
1: VNET ← IdentifyVNETs(Moving_Candidate_List)
2: for all vnet_k ∈ VNET do
3:    LaunchTimeoutToWaitForRequest(vnet_k, request_arrival[k] t)
4: end for
5: for index = 0 to k do
6:    if request_arrival[index] = true then
7:       neighbor ← GetSubstrateNeighborInfo(request_arrival[index])
8:       if ViabilityOfMigration(vnet_index, neighbor)= true then
9:          SendConfirmationOfMatch(neighbor, vnet_index)
10:         GetConfirmationToMigrate(neighbor, vnet_index)
11:         MigrateVirtualNode(neighbor, vnet_index)
12:         GetConfirmationVirtualNodeMigration(neighbor, vnet_index)
13:      end if
14:   end if
15: end for
```

The virtual manager executing the Algorithm 5.3, in a neighbor substrate node, also determines, in a parallel fashion, which are the virtual networks that need to have their virtual nodes migrated (line 1 of Algorithm 5.3) and then, waits a certain amount of time t for the arrival of requests to move virtual nodes associated to the virtual network (lines 2-4 of Algorithm 5.3). Once the match of the intentions of each neighbor is detected (line 9 of Algorithm 5.2 and line 8 of Algorithm 5.3), the preparations and handshakes of the migration process are executed by both virtual managers (*i.e.*, lines 10-12 of Algorithm 5.2, and lines 9-12 Algorithm 5.3).

An illustration of the parallel behavior associated to the execution of the self-organizing algorithm is presented in Figure 5.5, that is a partial view of the network showed in Figure 5.3. The internal state of the self-organizing algorithm at the end of the first stage is depicted in Figure 5.5(a). The links that are not considered overloaded ("SL#1", "SL#2", "SL#3", and "SL#6") are discarded from the analysis of the self-organizing algorithm, while the overloaded one, "SL#4", has its traffic investigated. When substrate node "B" analyzes the substrate link "SL#4", it discovers that this link is overloaded by the incoming traffic associated to the virtual pipe of VN1. Based on this, the self-organizing algorithm on substrate node "B" declares itself as a receiving candidate for a virtual node of VN1. During the first stage there is no communication between neighbors, and thus the self-organizing algorithm on substrate node "B" supposes that a virtual node of VN1 is deployed at substrate node "E" and then proceed to the next stages.

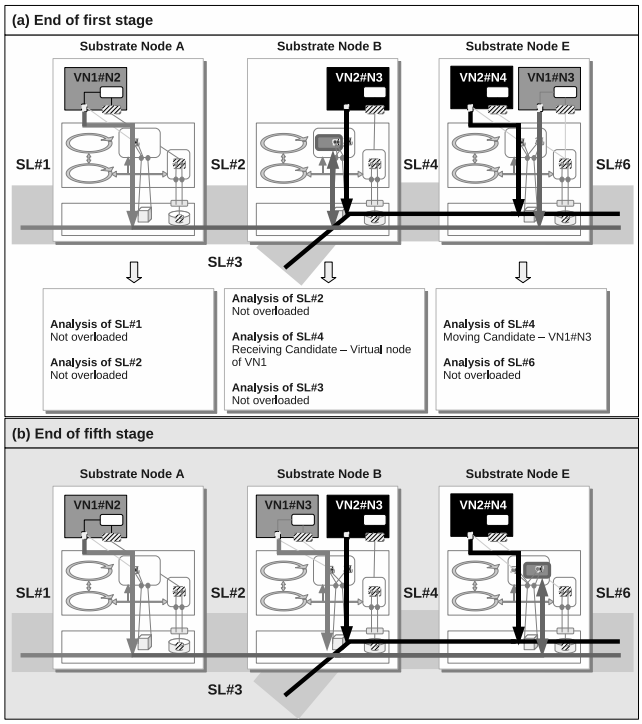


Figure 5.5: Execution of the self-organizing algorithm

In parallel, the substrate node "E" also analyzes and declares "SL#4" as overloaded link, and considers the outgoing traffic associated to "VN1#N3" as the traffic overloading this substrate link. In this case, the self-organizing algorithm on substrate node "E" considers "VN1#N3" as a moving candidate, and waits for a request to move a virtual node from VN1 coming from "SL#4". When this request arrives in substrate node "E", there

is a match between the virtual node that is requested to be moved (a virtual node of VN1) and the virtual node that should be moved from the substrate node "E" ("VN1#N3"). After this, the self-organizing algorithm inside the substrate nodes keeps executing until the last stage. The resulting environment is presented in Figure 5.5(b), where "VN1#N3" has been moved to the substrate node "B" and the traffic from "VN1#N3" to "VN1#N2", that was considered the cut-through traffic overloading "SL#4", is not there any more.

5.3.2 Receiving Candidate Heuristic

The receiving candidate heuristic identifies a cut-through traffic when a virtual pipe is associated with the overloaded virtual link vl_j^k inside the substrate link sl_i. This heuristic is based on the comparison of the incoming traffic against the outgoing traffic of a virtual pipe inside a substrate node. Three conditions must be followed in order to declare the substrate node as a receiving candidate of a virtual node.

- **Condition 1.** Guarantees that no read and write operations are associated to the virtual elements of the virtual network *k* under analysis on a substrate node.

$$NO\_InputOutput(k) = \begin{cases} 1 & \text{if } (\textit{virtual\_read}_k = 0) \vee (\textit{virtual\_write}_k = 0) \\ 0 & \text{otherwise} \end{cases}$$

- **Condition 2.** Identifies if the main traffic of virtual link vl_j^k of virtual network k is an incoming traffic.

$$IN\_MainTraffic(j,k) = \begin{cases} 1 & \text{if } IN(vl_j^k) \geq T(vl_j^k) - IN(vl_j^k), \\ 0 & \text{otherwise} \end{cases}$$

  where,
  IN(link) is a function that returns the incoming traffic, and
  T(link) is a function that returns the total traffic.

- **Condition 3:** Correlates the amount of incoming traffic of vl_j^k with the outgoing traffic (given by the function $OUT(link)$) of all the other virtual links of the same virtual network k. The set of virtual links belonging to a virtual network is denoted by $VirtualLinkVNET^k$. The goal is to detect whether the incoming traffic on vl_j^k is actually being forwarded through one or multiple distinct virtual links of the same virtual network.

$$OutgoingTraffic(j,k) = \begin{cases} 1 & \text{if } IN(vl_j^k) \leq \left(\sum_y OUT(vl_y^k)\right), \\ & \text{where } j \neq y \text{ and } vl_y^k \in VirtualLinkVNET^k \\ 0 & \text{otherwise} \end{cases}$$

The final analysis to determine if the virtual link has to be inserted in the *Receiving_Candidate_List* is presented in Equation 5.1. The tuple <*virtual network, virtual link*> characterizes the receiving list, where the first element of the tuple can have multiple entrances, but the second one is unique.

$$Receiving\_Candidate\_List = \begin{cases} < vnet_k, vl_j^k > & \text{if } \begin{pmatrix} NO\_InputOutput(k) \ \& \\ IN\_MainTraffic(j,k) \ \& \\ OutgoingTraffic(j,k) \end{pmatrix} \\ \emptyset & \text{otherwise} \end{cases} \tag{5.1}$$

5.3.3 Moving Candidate Heuristic

The moving candidate heuristic verifies if a virtual node is generating the outgoing traffic on vl_j^k. It is assumed that a virtual node uses not only virtual link resources to originate a network traffic in vl_j^k, but also uses IO capacity. For example, before transmitting packets, the application inside the virtual node needs to buffer these packets and for doing this it needs to use the slices of IO resources assigned to itself. The moving candidate heuristic identifies a relationship between the outgoing traffic of link vl_j^k with the amount of IO used by the virtual network (k), and the incoming and outgoing traffic associated to all the other virtual links of such virtual network. This relationship is established based on the conditions listed below.

- **Condition 1:** The virtual network *k* must perform read or write operations on the virtual IO slices.

$$IO\_Operation(k) = \begin{cases} 1 & \text{if } (\textit{virtual\_read}_k > 0) \vee (\textit{virtual\_write}_k > 0) \\ 0 & \text{otherwise} \end{cases}$$

- **Condition 2:** The outgoing traffic of vl_j^k must be higher than the incoming traffic of the same virtual link.

$$OUT\_MainTraffic(j,k) = \begin{cases} 1 & \text{if } OUT(vl_j^k) \geq T(vl_j^k) - OUT(vl_j^k), \\ 0 & \text{otherwise} \end{cases}$$

- **Condition 3:** The outgoing traffic of vl_j^k must be associated with an amount of data used by the read and write operations associated to the virtual network k. Due to the fact that any data packets of the virtual networks are inspect, it becomes necessary to establish an association between the outgoing traffic of vl_j^k and the IO resources consumed by the virtual node of the virtual network *k*. In this case study, this relationship is established considering the amount of data read from the virtual slices, expected to be sent through the virtual links. These relationship is characterized using similarity functions described below.

$$Similarity(j,k) = \begin{cases} 1 & \text{if } \textit{BottomSimilarity}(vl_j^k) \leq V(j,k) \leq \textit{UpSimilarity}(vl_j^k), \\ 0 & \text{otherwise} \end{cases}$$

where,
BottomSimilarity$(vl_j^k) = vl_j^k - (vl_j^k * \delta)$,
UpSimilarity$(vl_j^k) = vl_j^k + (vl_j^k * \delta)$,
$V(j,k) = \left(\left(\sum_y IN(vl_y^k)\right) + virtual\_read_k\right) - \left(\left(\sum_z OUT(vl_z^k)\right) + virtual\_write_k\right)$,
$\delta \in \Re : 0 \leq \delta \leq 1$, being δ a similarity factor,
$vl_y^k, vl_z^k \in VirtualLinkVNET^k$, and $y, z \neq j$.

The virtual node associated to the virtual network k whose virtual link vl_j^k is under evaluation, is declared a candidate to be moved according to the Equation 5.2. Indeed, the virtual link vl_j^k is inserted in the *Moving_Candidate_List* to be further analyzed by the other stages of self-organizing algorithm, as depicted before.

$$Receiving\_Candidate\_List = \begin{cases} < vnet_k, vl_j^k > & \text{if } \begin{pmatrix} IO\_Operation(k) \ \& \\ OUT\_MainTraffic(j,k) \ \& \\ Similarity(j,k) \end{pmatrix} \\ \\ \emptyset & \text{otherwise} \end{cases} \tag{5.2}$$

5.3.4 Implementation

Despite the potential to gain a large market-share, network virtualization imposes large enhancements on the current network models. Some of the limitations that need to be enhanced in such models are: new devices, transmissions mechanisms, isolation of information, etc. Given such limitations, a clean slate approach was identified as the best alternative for implementing and testing the proposed self-organizing P2P solution. Thus, a network virtualization module was developed based on Omnet++ simulator (OMNET++, 2010), using a packet-oriented transmission mechanism with traffic-shaping. The details of the network virtualization module are described in the Appendix B, while Figure 5.6 illustrates the developed environment in the context of the Omnet++ simulator.

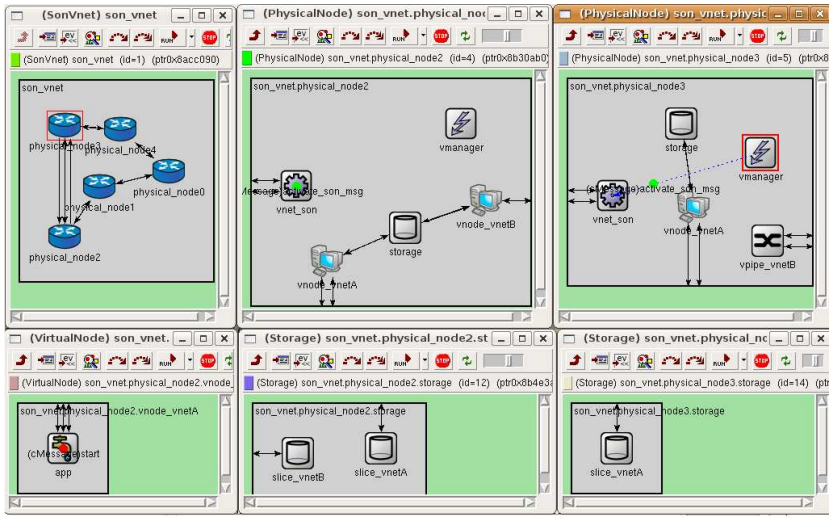


Figure 5.6: Snapshot of virtual environment deployed with Omnet++

One important aspect of the implementation is the monitoring process, since the self-organizing algorithm depends on this process to make decisions. A two-stage monitoring process was defined. The first stage is always active and the size of data passing through the measurement points is the monitoring information buffered. The second stage is periodically activated and for the experiments the interval of 1.5 minutes is used. The monitored information on the first-stage buffer is summed and stored in a second-stage buffer. The self-organizing control loop uses the information of the second stage to determine the average amount of resources consumed within two self-organizing cycles. A sliding window keeps part of the information of the second-stage and the data from the first-stage is always erased. The sliding window helps to avoid that punctual high loads trigger constant reorganizations and lead the substrate network to an unstable state. Further information about the monitoring process can be found in Appendix B.

5.4 Experimental Evaluation

The increasing demand of multimedia services over the Internet is pushing for new methods to allocate resources in future networks. For example, IPTV services are expected to become more and more popular and integrated offers, like the triple-pay packages (BALDI, 2006), requiring, thus, cost-effective strategies for resource allocation (GUNKEL et al., 2008) (HAN; LISLE; NEHIB, 2008). In fact, a typical IPTV network

infrastructure requires significant investments for the distribution network, in terms of guaranteed bandwidth as well as available storage capacity. Normally, these resources need to be planned and well dimensioned in advance, before upper services can be actually deployed (DEGRANDE et al., 2008).

The costs of deploying a physical infrastructure may prevent many service providers to get into the market, like in the case of IPTV services (HAN; LISLE; NEHIB, 2008). As discussed before, the key on network virtualization is of dividing the physical network infrastructure into several slices and associating them to different virtual providers. The needs of service providers, like IPTV providers, match perfectly with the nature of network virtualization environments. In this way, the evaluation of the self-organizing P2P approach is performed considering a network virtualization scenario composed of a substrate provider and virtual IPTV providers. The testbed and the results of the experiments are presented as follows.

5.4.1 Testbed

One of the architectures proposed for deploying IPTV services (DEGRANDE et al., 2008) is depicted in Figure 5.7. The Super Head-End (SHE) element is responsible for receiving and storing the flows, in this case TV channels and videos, from national content providers. These flows are forwarded through a core network infrastructure, and stored on the Video Hub Offices (VHO). Then the flows are transported over the metropolitan network and stored on the Video Serving Office (VSO) devices. Finally, the IPTV content reaches the home network and is delivered to the Set-Top Box (STB) inside the end user device. In this testbed, it is assumed that the SHE, VHO, and VSO are the elements virtualized and the self-organizing P2P approach is applied to manage the VSO elements.

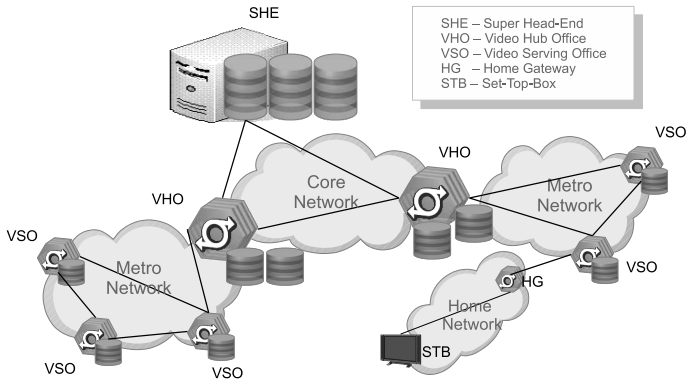


Figure 5.7: IPTV architecture

The testbed is composed of two IPTV providers covering the same metropolitan region. Network topologies and the initial mapping of the virtual IPTV networks are depicted in Figure 5.8. The substrate network is composed of 9 substrate nodes and each virtual IPTV network is composed of 3 virtual nodes (physically separated by 2 virtual pipes). Each virtual node in Figure 5.8 is an IPTV VSO (DEGRANDE et al., 2008) able to store and transmit movies.

Two sets of flows are running inside the substrate network. The first one is associated to user's requests of the Virtual Network 1 (VN1). For VN1, the users connected to the VSO "VN1#N1" (inside node "B") are requesting movies that are associated to "VN1#N3" (inside node "H"). The movies are transmitted over the virtual link

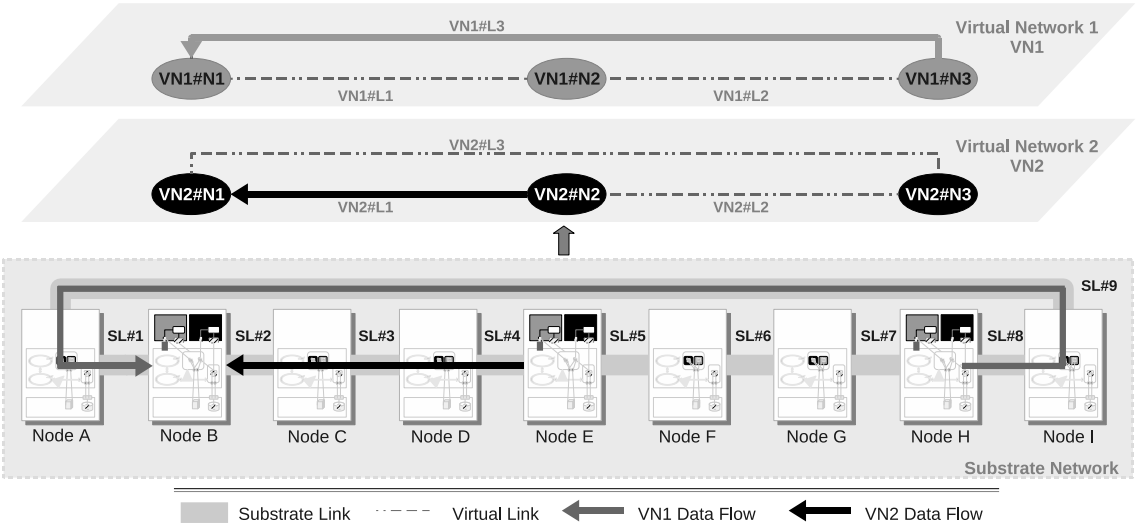


Figure 5.8: Evaluation scenario

"VN1#L3", which is mapped to three substrate links "SL#1", "SL#8", and "SL#9". As depicted in Figure 5.8, the flows of VN1 start at node "H" and arrive at the node "B". The second set of flows is associated to the Virtual Network 2 (VN2). The users connected to the VSO "VN2#N1" (inside node "B") are requesting movies that are stored at "VN2#N2" (inside node "E"). The transmission occurs through the virtual link "VN2#L1", which is mapped to substrate links "SL#2", "SL#3", and "SL#4". The flows of VN2 are originated at node "E" and arrive at node "B".

The bandwidth of each virtual link is 500 Mbits while the bandwidth of each substrate link is 1 Gbits. The size of the virtual storage associated to the VSOs of the virtual nodes is 50 GB, and the storage capacity of each substrate node is 100 GB. The size of the packets to be transmitted in the substrate links is fixed to 1 MB. The threshold to identify an overloaded link is the equivalent to 60% of the virtual link bandwidth.

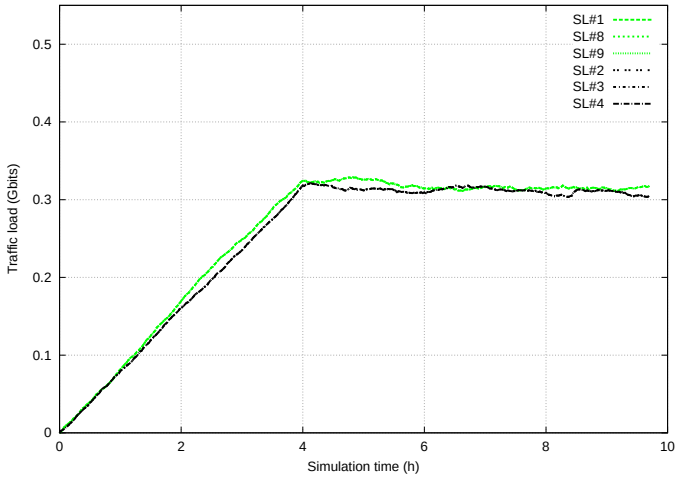
The amount of traffic inside virtual networks is mainly influenced by the number of movies requested by the users of the virtual networks. For this scenario, the request rate for each virtual network is fixed to 400 requests of movies per hour (400 req/h), and the interval between each request is given by an exponential distribution (this request model is based on (AGRAWAL et al., 2007)). The request rate is kept constant and active during the whole simulation. When a request arrives, the next action is the transmission of the movie. All movies in the experiments have the same size (4 GB).

The evaluation shows the efficiency of the self-organizing P2P approach in terms of spared network traffic. Almost 10 hours of user's request were simulated, being the self-organizing cycle activated every 5 minutes. Traffic load of the substrate links and the average latency of the packets are measured every monitoring interval of 1.5 minutes.

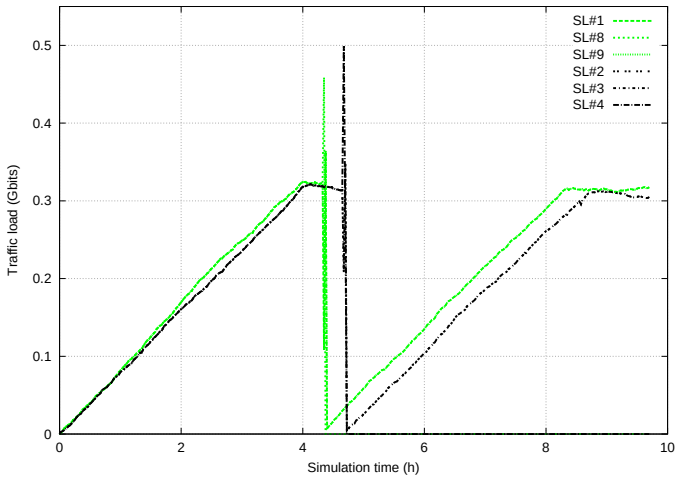
5.4.2 Network Traffic Load Analysis

The traffic load curves presented in the experiments show an ascending behavior until 4 hours of simulation, and after this the traffic load reaches the maximum average load per virtual link for the request rate of 400 req/h, i.e., around 0.325 Gbits.

During the simulations with the self-organizing model disabled, Figure 5.9(a), the substrate links used by VN1 (light-lines) have their curves overlapped because they have



(a) Self-organizing disabled



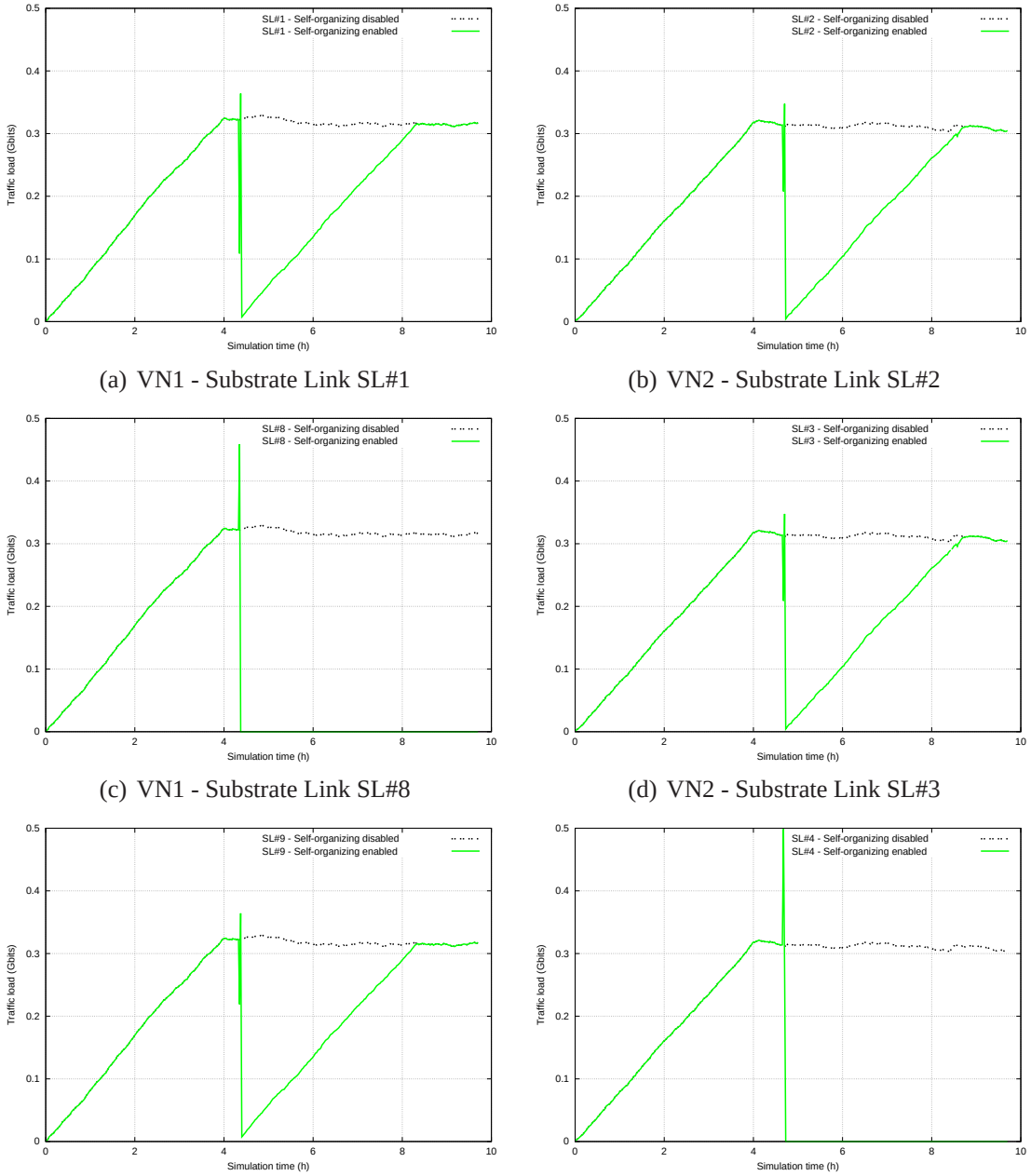
(b) Self-organizing enabled

Figure 5.9: Traffic load of all substrate links

the same traffic load. The same overlapping occurs with the traffic load of the substrate links used by VN2 (dark-lines in Figure 5.9(a)). The traffic of both virtual networks follows a cut-through pattern and for this reason the traffic load of those substrate links is the same. However, when the self-organizing model is enabled, Figure 5.9(b), the traffic load of some substrate links changes. The self-organizing model reallocates the virtual resources in two distinct cycles. The first one happened after 4 hours of simulation and reorganized the virtual resources of VN1. The second cycle happened near 5 hours of simulation, and the virtual resources of VN2 were reorganized.

To understand how the self-organizing model changes the traffic load of the substrate links we present Figure 5.10, that shows the traffic load of each substrate link when the self-organizing model is enabled and disabled. The traffic load of each substrate link associated to the VN1 is presented in Figures 5.10(a), 5.10(c), and 5.10(e), while the traffic load of the VN2 substrate links is illustrate in Figures 5.10(b), 5.10(d), and 5.10(f). Analyzing the traffic load of the substrate links presented in Figure 5.10 it is possible to discover which virtual nodes moved during the self-organizing cycles.

In the case of the VN1, the traffic load of substrate link “SL#8”, Figure 5.10(c), indi-



(a) VN1 - Substrate Link SL#1

(b) VN2 - Substrate Link SL#2

(c) VN1 - Substrate Link SL#8

(d) VN2 - Substrate Link SL#3

(e) VN1 - Substrate Link SL#9

(f) VN2 - Substrate Link SL#4

Figure 5.10: Traffic load of each substrate link used by the testbed

cates that virtual node “VN1#N3” moved from substrate node “H” to “I”, because after 4 hours of simulation the traffic load changes from 0.32 Gbits to 0.46 Gbits and right after this it is interrupted (drops to zero). Associated to this fact, the traffic load of substrate links “SL#1”, Figure 5.10(a), and “SL#9”, Figure 5.10(e), also increases from 0.32 Gbits to 0.36 Gbits after 4 hours of simulation. This increase occurs because the virtual node “VN1#N3”, now placed at the substrate node “I”, processes all packets queued during the moving phase. In the sequence, the traffic load of “SL#1” and “SL#9” drops to 7.4 Mbits and starts to rise again as soon as the virtual application inside the virtual node “VN1#N3” restarts to transmit movies in response to new requests. Progressively, the traffic load of those substrate links increases until it reaches the maximum average load (0.325 Gbits)

after 8 hours of simulation. The same traffic behavior is observed for the VN2. In this case, the virtual node “VN2#N2” moved from the substrate node “E” to “D”, and this caused the elimination of traffic load on the substrate link “SL#4”, Figure 5.10(f). The pattern of increasing, dropping, and increasing again is also observed in traffic load of the substrate links “SL#2” and “SL#3”, respectively in Figure 5.10(b) and Figure 5.10(d).

The graphics of Figure 5.10 show that using the self-organizing model it is possible to reduce 1/3 of the traffic load when the transmissions on substrate links “SL#8” and “SL#4” are eliminated. Now, to have an entire picture of the amount of network resources spared with the self-organizing model, we present Figure 5.11, where the traffic load of all substrate nodes of the scenario is summed and graphically presented. Considering the scenario used on the evaluation, the total traffic load of the network is approximately 1.9 Gbits when the self-organizing model is disabled, and when the model is enabled it reaches the stable state using 1.2 Gbits. This means that 36.8% of the network resources of this scenario were spared when the self-organizing model is applied to managed the network resources.

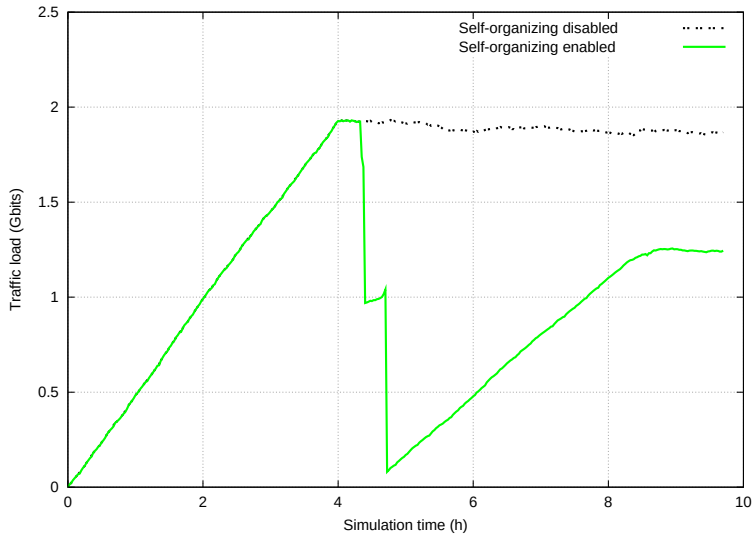


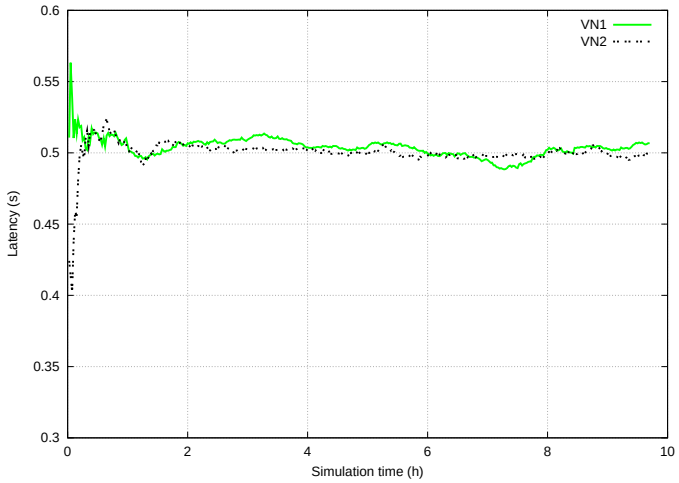
Figure 5.11: Sum of traffic load from all substrate nodes

Despite the advantages on using the self-organizing model to managed the traffic load of the substrate network, there are issues regarding at least (i) the interruption time of the applications inside the virtual networks and (ii) the latency of the packets when a virtual node migration is required during the self-organizing cycle. The first issue mainly depends on the efficiency of the moving mechanism and the amount of data that has to be moved. The second issue this investigated as follows.

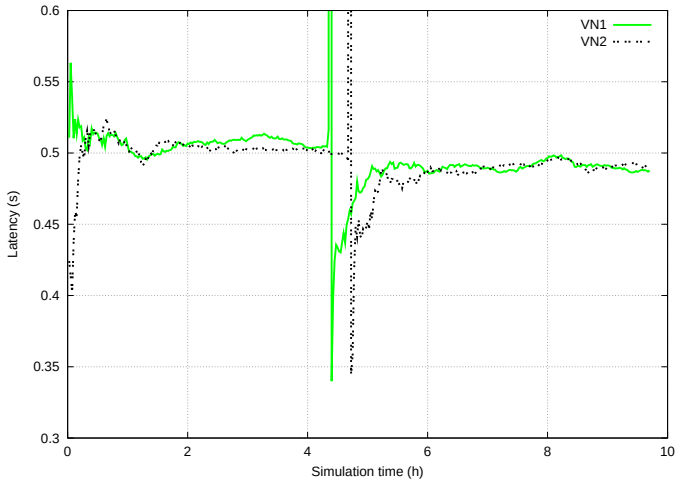
5.4.3 Packet Latency Analysis

This experiment shows the average latency of packets arrived in the destination virtual node of each virtual network within the monitoring interval (1.5 minutes). Figure 5.12 shows the packet latency measured for each virtual network.

The latency of the packets for both virtual network is approximately 0.51s when the self-organizing model is disabled, as illustrated in Figure 5.12(a). The same average latency is observed in Figure 5.12(b) until the moment that the first reorganization is executed. It is not illustrated in Figure 5.12(b), but the average latency during the period



(a) Self-organizing disabled



(b) Self-organizing enabled

Figure 5.12: Latency of virtual network flows

of reorganization associated to the VN1 reaches 50.04s, and during the reorganization related to VN2 it is approximately 50.84s. After such reorganization, the average latency remains stable around 0.48s for both virtual networks. This value represents a reduction of 5.9% of the average latency if compared to the latency before the reorganization.

The self-organizing model can reduce the latency of the packets but the cost associated to this benefit is a high latency (around 50s), as presented in Figure 5.12(b). This high latency might become a problem when the applications running inside the virtual network require strict QoS (Quality of Service) guarantees.

5.5 Critical Evaluation of the Designed Approach

The final analysis of the self-organizing P2P approach is presented here in this section. First, the compliance to the management requirements is discussed. Second, the degrees of accomplishment of the integration issues are identified, and finally, the potentialities and drawbacks of such approach are listed and examined.

5.5.1 Compliance to Management Requirements

Likewise the analysis done for the previous case study, this section depicts the discussion of the accomplishment of the management requirements on the self-organizing P2P solution developed in this chapter. A summary of such discussion is presented in Table 5.1 and the details are described below.

The achievement of the efficient use of resources is twofold. First, the monitoring process does not need to inspect the network and IO packets of the virtual elements. Thus, the consumed processing power is related to the act of recognizing the size of the packets and not the content of such packets. Doing this, the virtual manager (*i.e.*, the element that intermediates the substrate and virtual resources) does not stall operations associated with the packets, like receiving/sending. Second, the communication to organize the relocation of virtual resources is reduced to substrate neighbors. This means that messages are exchanged in local areas of the substrate network and not in a global scale.

Table 5.1: Achievement of management requirements on self-organizing P2P case study

Requirements		**Accomplishment**
Efficient Resource Usage		Monitoring process without inspection of packets.
		Few management messages exchanged between neighbors willing to reorganize the virtual resources.
Agile Management Actions		No human intervention is required to execute the self-organizing process.
Transparency	Access	-
	Location	The placement of the virtual elements is hidden.
	Migration	Virtual elements can be moved without the awareness of the users.
	Relocation	During the relocation of resources the users can access but the response time rises.
	Replication	-
	Concurrency	-
	Failure	-
Parallel and Simultaneous Behavior		Relocations occur in different parts of the substrate network at any time.
		Horizontal parallelization is accomplished by the match on the needs of neighbors.

Agile management actions are achieved in the proposed self-organizing P2P solution once there is no human intervention to identify bottlenecks and change the environment in order to better use the resources (*i.e.*, in this case study the network bandwidth). The proposed solution is able to identify virtual elements that should be reorganized and act by itself in order to reduce the traffic consumption on the substrate network perspective.

The compliance to transparent management actions is divided into seven types of transparency, as exposed by Table 5.1. Access and concurrency types of transparencies are part of the virtualized environment. However, they are not explicitly part of the proposed solution in this case study, and for this reason they are out of the scope of this

analysis. The replication and failure transparencies are not consider either on the virtualized environment of this case study or on the proposed solution. Thus, they are also out of the analysis scope. Finally, three types of transparencies are relevant for the self-organizing P2P solution: location, migration, and relocation. The users of the applications running inside the virtual networks are not aware of where exactly those applications are placed, which constitute a location transparency. In addition, when a reorganization of the virtual elements is required, the users of such virtual elements have no knowledge about the fact that the virtual resources are being migrated. Finally, the self-organizing P2P solution provides partial compliance to relocation transparency, because during the relocation of the resources the users can access them but a difference on response time will be experienced.

Parallel and simultaneous behavior is also a management requirement accomplished by the self-organizing P2P solution. The request and match algorithm used by the solution described in this chapter is the key for the accomplishment of this management requirement. The decentralization of the decision of when it is necessary to reorganize the virtual resources and the local communication between neighbors enable self-organizing actions at the same time in different parts of the substrate network. The horizontal parallelization is native to the self-organizing algorithm running inside each substrate node. No external entity is needed to activate the reorganization and the trigger is the request and match of the neighbors' needs.

Summarizing, this section discussed how the self-organizing P2P solution is able to be compliant to the management requirements established in this thesis. The next section extends the critical evaluation of the developed solution by presenting the analysis of integration issues.

5.5.2 Achievement of Integration Issues

Once again, the characteristics of the integration issues defined in this thesis are compared against the design features of the self-P2P approach developed in this case study. Table 5.2 shows the result of such comparison.

Different from the what happened on the evaluation of the design of the self-healing P2P approach (Section 4.4.2), the self-organizing P2P approach achieved the highest accomplishment degree of integration in all issues, except the definition of light self-elements. This success is, in a greater part, attributed to the nature of the self-* property and to the design of a the self-element associated to this property.

In this case study, the self-organizing property is designed as an indivisible self-element. The advantages can be clearly observed in Table 5.2, where common knowledge of management task, local information, parallel and distributed algorithms for decision-making, and reduction of explicit and global coordination were achieved. In contrast, not achieving the design of light self-elements could be understood, in a first sight, as a negative aspect associated with the design of a unique self-element to perform the tasks of a self-* property. However, it is the exactly inverse interpretation that has to be read from this situation.

The design of an unique self-element that executes all the attributions of the self-organizing property enabled the employment of techniques from the parallel and distributed area, such as the SPMD model of designing algorithms (MURESANO; REXACHS; LUQUE, 2009). The combination of a single program (*i.e.*, common knowledge of the management task) executed over multiple data (*i.e.*, local information) with the idea

Table 5.2: Degree of integration issues for self-organizing P2P approach

Integration Issue	Accomplishment Degree	Justification
Common knowledge of management task	Achieved	The same monitoring and self-organizing algorithms are executed by all management entities
Local information	Achieved	The information that starts the self-organizing process is derived from the analysis of local information retrieved by the monitoring control loop.
Parallel and distributed algorithms for decision-making	Achieved	Self-organizing algorithm is parallel following the Single Program Multiple Data (SPMD) model.
		The decision-making employs a matching process between complementary perspectives identified by distinct management entities.
Light self-elements	Not Achieved	Complementary perspectives inside the same self-element.
		Requires the definition of heuristics to analyze the local information.
		Algorithm requires control mechanisms to handle asynchronous operations.
Reduction of explicit and global coordination	Achieved	Communications is restricted to neighbors.
		There is an implicit coordination mechanism activated by the result of the local analysis.

of P2P interactions can result in highly distributed execution of self-* properties. The major advantage of the proposed unique self-element is its capacity to infer that a neighbor should be suffering from complementary effects of the locally identified situation with minor explicit communication and no human intervention.

5.5.3 Potentialities and Shortcomings

Despite the advantages identified before, the self-organizing P2P approach offers another important potentiality: its capacity of being easily adapted to other management scenarios. Once the heuristic of the self-organizing approach is changed, it is possible to re-use the same logic. For example, consider a network management platform based on a P2P overlay, where management services (inside peers spread along the management

overlay) are used to get in contact with managed entities. The initial placement of the management services inside the overlay is given, typically, by the network administrator. However, if over time a management service starts to retrieve a considerable amount of information and it is physically distant from the managed entity, the self-organizing P2P approach could be used to approximate the management service and the managed entity. The objective would still be sparing network bandwidth of the physical infrastructure where this network management P2P overlay is placed.

In this scenario, instead of moving the source of the traffic (the managed entity) only the destination of the traffic (the management service) could be moved. In this sense, it becomes necessary to change the heuristics of the first stage that define which peer receive and which one moves the resources (in this case, the management service). Thus, no logical changes are required on the other 4 stages of the self-organizing P2P approach but, indeed, changes on the implementation of the self-organizing algorithm are necessary to adapt to the type of resource being moved.

In essence, the correct employment of the self-organizing P2P approach depends on the existence of underlaying infrastructure and an overlay whose components are compliant with the concept of being replaced in different parts of the underlaying infrastructure. One major issue associated to this dependency, is the transparency between the overlay and the underlaying infrastructure. No transparency could lead to a plain infrastructure, where no abstraction between overlay and underlay elements would be provided. On the other hand, too much transparency could lead to situations where moving the resource would be impossible because the effects of the migration (*e.g*, interruption time) would break the transparency in the overlay perspective. Therefore, the transparency issue can become a shortcoming on the employment of the self-organizing P2P approach.

Another issue associated with such approach that can become a shortcoming is the definition of fine-grained costs to determine whether a change on the placement of the resources is really profitable for the managed network infrastructure. Very strict cost evaluations could lead to the stagnation of the network, *i.e.*, the resources could never be moved and thus the network resources could not be used in a more proper way. In contrast, very tolerant and flexible cost evaluations can lead to instability of the system, once reorganizations could become too frequent.

5.6 Final Remarks

This chapter presented the design, instantiation, and evaluation of the self-organizing P2P approach. The self-organizing property is used to reorganize elements of the network so that the traffic inside the substrate network links can be reduced. The P2P employment regards more to designing interactions among the entities of the network management environment than to defining an infrastructure (*i.e.*, overlays). The key behind the employment of P2P interactions is to investigate the development of P2P network management applications, instead of solely exploring P2P infrastructures for network management (as detailed in Section 2.3).

The union of self-organizing property and P2P interactions resulted in a self-organizing P2P approach that is able to reorganize the elements of the network during their lifetime. This approach can be employed if at least these conditions are followed: (i) the elements that need to be moved are generating/receiving traffic; (ii) there is a cut-through traffic inside the network; (iii) there is an abstraction between the underlaying network and the infrastructure where the elements to be moved are running; and (iv) it is possible to iden-

tify complementary perspectives for a single event that can happen to the elements to be reorganized (such perspectives are used to define the moving and receiving candidates elements of the network).

In fact, complementary perspectives are a key issue on the design of the self-organizing P2P solution, because they create the conditions to develop the intentional cooperative posture. The idea of sharing the responsibility of the decision-making, among entities that want to move resources and the ones that want to receive resources, form the basis for constructing the following methods of cooperation: grouping, communication, collaborating by sharing information and decisions, and coordination of tasks.

Each match on the desires of the cooperative entities (*i.e.*, moving and receiving candidates) constitute a small group of two elements. After sharing information, these entities agree on the decision of the next steps and coordinate their tasks in order to accomplish the migration of resources. This way, the cooperative behavior is embedded into the idea of complementary perspectives mechanism, being a native aspect of the applications developed based on this mechanism.

In addition to the remarks depicted above, this chapter also presented the employment of the self-organizing P2P solution on the scenario of resource management on network virtualization. The problem of this scenario is to manage the traffic consumption on the substrate network during the lifetime of the virtual networks deployed on top of such substrate network. This case study matches all the conditions for the employment of the self-organizing P2P approach, thus, the instantiation of such approach was evaluated. The experiments showed that the benefits in terms of reduction of traffic load are more expressive than the results in terms of latency. Indeed, the high latency observed during the reorganization process is an important aspect that has to be observed, because it can become an issue that prevents the employment of the self-organizing P2P approach in different scenarios.

Finally, the self-organizing P2P approach achieved the employment of common knowledge of management task, local information, parallel and distributed algorithms for decision-making, and the reduction of explicit and global coordination. The key for achieving those integration issues is the direct mapping of the self-organizing property to a unique self-element.

6 RESULTS DISCUSSION

The purpose of this chapter is to present the observation and interpretation of results derived from the analysis of subjective aspects of the case studies. The first topic analyzed is associated with the outcomes that can be learned from the observation of the self-* P2P approach under the perspective of the parallel and distributed computing research area. In the sequence, the design of the integration issues in both case studies are examined. The major outcome of this examination is that (in the context of network management research) it is possible to identify different dimensions associated with the self-* P2P approach, and this discussion is presented in a dedicated section. Finally, this chapter also establishes how the self-* P2P approach fits in the network management scenario.

6.1 Self-* P2P in the Light of Parallel and Distributed Computing

One of the features explored by the self-* P2P approach for network management is the capacity of providing the ways to define management applications that execute parallel and distributed actions. Thus, this section analysis the self-* P2P approach taking into account the basis of Parallel and Distributed Computing (PDC) research area. The basic concepts of PDC used for the purpose of this analysis follow the definition presented by Zomaya (ZOMAYA, 1996), and are briefly reviewed in the next subsection.

6.1.1 Transposing Parallel Paradigms to Network Management Approaches

Flynn (ZOMAYA, 1996) defined one of the most accepted classifications of computer architectures, and his criteria was based on how the machine relates its instructions to the data being processed. This classification is composed of four classes: Single Instruction Stream, Single Data Stream (SISD); Single Instruction Stream, Multiple Data Stream (SIMD); Multiple Instruction Stream, Single Data Stream (MISD); and Multiple Instruction Stream, Multiple Data Stream (MIMD).

There is a very strong relationship between the computer architecture and the algorithms developed to run on top of this architecture. According to Zomaya: (i) sequential algorithms are associated with SISD class; (ii) sequential algorithms can be transformed to deal with parallel data stream in the case of SIMD architectures - which means that the parallelism is given by the data and not by the logic of the algorithm; (iii) parallel algorithms based on the execution of different operations can be developed to run on MISD architectures; and (iv) fully parallel algorithms can be developed when the MIMD architecture is considered.

Based on the aforementioned, Table 6.1 presents an analysis of how the parallel processing paradigms can be transposed to network management approaches. The objective is to show how the transposition helps to identify which are the classes of algorithms associated with each one of the common network management approaches and also with the self-* P2P approach proposed in this thesis.

Table 6.1: Analysis of parallel paradigms and network management approaches

Approaches x Paradigms	**Centralized**	**Hierarchical**	**Self-* P2P**
SISD	√		
SIMD		√	
MISD		√	
MIMD			√

In order to make the transposition of parallel processing paradigms to network management approaches it is necessary to base the analysis in some assumptions that are described as follows. The manager entity on network management approaches is equivalent to the control unity on the parallel processing paradigms. Likewise, the agents are mapped to the processing units, and the memory is the network itself considering the visible domain of the manager entity. Finally the instruction flow is mapped to the actions of the manager entity, while the data flow is mapped to the managed devices/services.

Having the aforementioned assumptions in mind, it is possible to verify that centralized solutions are in essence sequential algorithms executed by the central manager and for this reason they can be associated with SISD paradigm, as illustrated in Table 6.1. Hierarchical approaches can be either associated with SIMD or MISD paradigms of developing algorithms. In the case of SIMD paradigm, the hierarchical network management solution replicates managers that execute the same tasks in different parts of the network. In contrast, hierarchical approaches following an MISD perspective on the development of management algorithms, are based on the distribution of different tasks among the managers of the hierarchy. Thus, it is possible to say that the hierarchical approaches have their distribution focused by means of manager replication (SIMD) or by means of task replication (MISD), but the algorithm itself does not explore parallel and concurrent capabilities. Different from the traditional network management approaches, the proposed self-* P2P approach can be associated with algorithms that follow a MIMD classification. The proposed approach can benefit from both manager and task replication, and in addition can employ attributes of concurrent models (discussed in the next section) to turn the solution into a fully distributed and cooperative environment.

6.1.2 Relating Integration Issues and Attributes of Concurrent Models

The self-* P2P approach is envisioned to be a fully distributed and cooperative solution for network and services management. The core of such approach is to explore the development of more sophisticated management applications that use parallel, distributed, and concurrent features of the environment in which they are embedded. Therefore, the integration issues act as a guideline to achieve the development of the desired sophisticated applications. In fact, the integration issues are strongly related to models of concurrent programming.

According to Zomaya (ZOMAYA, 1996), concurrent programming is associated with several challenges on technical problems and also on the diversity of interpretations that it can lead. In order to enable the analysis of concurrent programming models, the author proposed a number of attributes that can characterize and differentiate these models. The list of the attributes of concurrency model are briefly described below.

1. Level of granularity - Defines which is the level of the observation of the parallel actions. It is also called level of atomicity.

2. Sharing the clock - It is related to a fine-grain level of concurrency where several processes share or not the same clock.

3. Sharing the memory - Regards to implications such as: space that may be used for sharing information; issues associated with the mutual or exclusive access to the information; and level of granularity.

4. Pattern of interaction - Defines the forms that processes can interact.

    4.1 Synchronization - Establishes a chronological order between events taking place in different processes.

    4.2 Communication - Defines a transfer of information between processes.

5. Pattern of synchronization - Establishes the manner that two process can interact when needing to share something in common.

    5.1 Mutual exclusion - Encapsulated sequence of actions whose execution is indivisible and associated to a single process.

    5.2 Mutual admission - Encapsulated sequence of actions that can be performed simultaneously by more than once process.

6. Specifying concurrency - Identifies the types of concurrent programming.

    6.1 Application - The problem or the situation is intrinsically concurrent and the solution explicitly considers the concurrency.

    6.2 Implementation - The problem or situation is intrinsically sequential and the concurrency is introduced in the implementation for efficiency reasons.

7. Pattern of communication - Defines how processes communicate with each other.

    7.1 Synchronous - Information is transfered when processes have undivided attention to the operation.

    7.2 Asynchronous - Information is transfered in two steps: the sender deposits the information in some area, then the receiver retrieves the message.

8. Implementation of concurrency - Identifies the ways of implementing an intrinsically concurrent problem.

    8.1 Effective - The concurrent processes are executed in different processors and interact in real time.

8.2 Simulated - The concurrent processes are executed sequentially on a single processor, and concurrency is simulated by the scheduling mechanism of the shared processor.

9. Interaction Protocol - Defines the types of protocols of concurrent processes.

9.1 Cooperative - Interacting processes know each other mutually and fulfill complementary functions to serve a common objective or interest.

9.2 Competition - Interacting processes not necessarily know each other; they fulfill distinct functions and typically (but not always) serve different interests.

Analyzing the description of the attributes and the proposed integration issues it is possible to establish a relationship among them, and this relationship helps to define which are the concurrent attributes that characterize the model of concurrency programing of the self-* P2P approach. Table 6.2 illustrates the identified relationship (the numbers used on the left column of the table are associated with the list of the attributes described above).

Table 6.2: Establishing the relationship of integration issues with the attributes of concurrent models

Integration Issues	Common knowledge of the management task	Local information	Parallel and distributed algorithms for decision-making	Light self-elements	Reduction of explicit and global coordination
1			√	√	
2					
3					
4.1	√	√	√		
4.2	√		√	√	
5.1					
5.2	√		√		
6.1	√	√	√		
6.2					
7.1	√		√		
7.2	√		√		
8.1			√	√	√
8.2					
9.1	√		√	√	√
9.2					

A first look in Table 6.2 indicates that almost all attributes of concurrency models are considered on the integration issues. The attributes 2 (sharing clock), 3 (sharing memory),

and 8.2 (simulated implementation of concurrency) are associated with fine grain aspects of concurrency that were not meant to be considered in the integration issues.

The attribute 5.1 (mutual exclusion pattern of synchronization) was intentionally not desired on the integration issues because it can reduce the parallelism and concurrency levels of the solutions that can benefit from a self-* P2P approach. In addition, the communicating processes are, in general, not in the same machine, therefore racing conditions and deadlocks on local information should not occur. Looking to the system level, implementation issues related to deadlocks must be treated by mutual exclusion pattern, but this is an specific situation and thus must not be intrinsic to the integration issues.

Intentionally, the integration issues were defined to privilege the specification of the network management solution in a parallel and concurrent manner. For this reason, the attribute 6.2 (concurrency specification on the implementation level) was left outside of the integration issues. This does not mean that implementation specification should not be treated when the employment of the self-* P2P approach, but it does mean that this is a low level issue and not a design issue.

The last attribute not contemplated on the integration issues is the 9.2 (competition type of interaction protocol). The idea on the integration issues of self-* properties and P2P is to enable the cooperation among the self-elements and not to encourage the competition. For this reason, the integration issues explicitly avoid the competition attribute.

Finally, the match between the other concurrency attributes and the integration issues on the design of self-* P2P solutions proves the strong tendency of the self-* P2P approach to employ distributed, parallel, concurrent, and cooperative mechanisms to the development of management applications since its conception phase. A detailed analysis of the integration issues is discussed in the next section with the objective of showing which are the contributions of those issues for the design of network management applications.

6.2 Analyzing the Design of the Integration Issues

The analysis of achieving the integration issues of each case study gave an individual perspective of the choices done to design the self-healing and self-organizing P2P solutions. In contrast, this section explores the direct comparison of both designs in the light of the integration issues, and Table 6.3 depicts such comparison.

Table 6.3: Comparison of achievement of integration issues

Integration Issues	Self-healing P2P	Self-organizing P2P
Common knowledge of management task	Partially achieved	Achieved
Local information	Not achieved	Achieved
Parallel and distributed algorithms for decision-making	Partially Achieved	Achieved
Light self-elements	Achieved	Not Achieved
Reduction of explicit and global coordination	Partially Achieved	Achieved

A simplistic look at Table 6.3 could lead to the interpretation that the self-organizing P2P solution is better than the self-healing P2P solution, once the first achieved the design

of a higher number of integration issues. Nevertheless, this is not the type of analysis that must be retrieved from this table, because it does not take into account the fact that both solutions achieved the development of the management requirements (as presented in Tables 4.3 and 5.1, respectively, from Sections 4.4.1 and 5.5.1). Thus, the correct interpretation of this table has to: (i) consider that both solutions are suitable for the management scenarios that they were designed for, and (ii) take a special look at the design decisions of the self-elements of each solution. As discussed in each case study, the decisions on designing or not light self-elements has a major influence on the other design issues. In order to compare, side by side, the major design features of each case study, Table 6.4 shows which are the differences introduced by the achievement or not of light self-elements on the self-healing and self-organizing P2P solutions.

Table 6.4: Analysis of the features of the designed self-* properties

Features to provide the self-* property	**Self-healing P2P**	**Self-organizing P2P**
Type of self-* property	Divisible	Indivisible
Perspectives of the problem	Partial to each type of self-element	Complementary and inside every self-element
Simultaneity capability of the strategy	Intrinsic when analyzed under the detection perspective	Intrinsic to each self-element
	Sequential when analyzed under the self-elements interaction perspective	
Communication interaction to solve the problem	Group-based	Neighbor-based

There is a clear distinction on the strategies of each case study. The first one - *i.e.*, self-healing P2P - uses a more distributed strategy to solve the problem of healing the system. Groups and self-elements with distinct roles are orchestrated in order to provide the self-healing property. In contrast, self-organizing P2P focuses on an individualized strategy with interaction based on small groups formed by neighbors to provide the self-* property. Although these strategies are very different on their form, the final objective of designing a fully distributed and cooperative management solution was achieved by both strategies. The analysis of the methods of cooperation (according to Jacques Ferber (FERBER, 1999).) employed by each strategy is summarized in Table 6.5.

Examining solely the comparison of the methods of cooperation employed by both self-healing and self-organizing P2P approaches, one could say that they are very similar. This similarity, on the other hand, could be understood as a common strategy of cooperative behavior. However, as proved by Table 6.4, the strategies or the forms of the design of the solutions are very different on their essence. This inconsistence between what can be expected from the methods of cooperation and the evidenced forms of the cooperative design leads to the conclusion that there might exist hidden relationships among the integration issues, features to provide the self-* property, and the cooperation methods. The next section shows the analysis of these relationships.

Table 6.5: Comparison of cooperative methods of the designed self-* properties

Methods of cooperation	Self-healing P2P	Self-organizing P2P
Grouping	√	√
Communication	√	√
Specialization	√	-
Collaborating by sharing tasks and resources	√	√
Coordination of actions	√	√
Conflict resolution by arbitration and negotiation	-	-

6.3 Delineating Dimensions of the Self-* P2P Approach

The inconsistency of the conclusions that can be derived from an isolated and individual analysis of the relationship of the integration issues, features to provide the self-* property, and the methods of cooperation rises the necessity of a deeper analysis of what is behind all these conclusions. The start point of this fine-grained examination is the fact that both divisible and indivisible types of self-properties (Table 6.4) can be designed with almost the same methods of cooperation (Table 6.5). However, the features to provide the self-* property are completely different.

The conclusion retrieved from these facts is that there exists different degrees of cooperation associated with the self-* P2P approach proposed in this thesis. Moreover, these degrees are intimately related with the nature of the self-* property, *i.e.*, whether it can be divided in multiples self-elements or if it is indivisible and provided by a single self-element. Based on this, it was possible to identify two dimensions associated with the self-* P2P approach: the **self-* property nature** and the **cooperative degree** dimensions. Figure 6.1 illustrates those dimensions while Figure 6.2 clarifies their relationship.

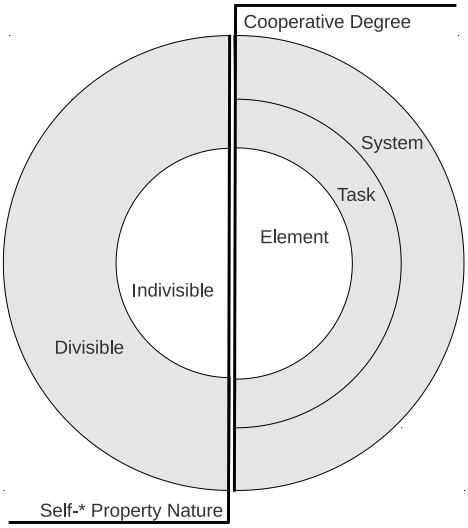


Figure 6.1: Dimensions of the self-* P2P approach

The self-* property nature dimension is composed of divisible and indivisible categories, which are defined according to the number of different types of self-elements that are required to provide the self-* property. The cooperative degree dimension is composed of three categories as illustrated in Figure 6.1. The criteria to define the cooperative degrees are based on the self-* property nature and how the logic to provide the self-* property is embedded inside the self-element(s) associated with the self-* property.

Observing Table 6.4 it was possible to derive that indivisible self-* properties are able to enclose in a single type of self-element the complementary perspectives of the same problem addressed by the self-* property. This capacity gives to the self-element the intrinsic simultaneity capacity of solving problems along the environment using for this a neighbor-based communication interaction. These features constitute, in fact, the **element** cooperative degree.

The analysis of divisible self-* properties and the examination of Table 6.4 resulted in the definition of other two cooperative degrees. The **task** cooperative degree is evidenced when multiples self-elements, belonging to the same self-* property, contain the partial view of the task to be performed by this self-* property. In this case, the task logic is not embedded inside all the self-elements, and thus these elements need to coordinate their actions, share information, and decisions based on their partial perspectives of the problem/task to be solved/performed.

The **system** cooperative degree is associated with the use of the logic of different types of self-elements belonging to different types of self-* properties in order to execute some task. The logic of the cooperation is not inside a self-element neither on a single self-* property, but it is associated with the orchestration of different independent tasks. This kind of cooperation degree also implies the design of highly modular self-elements so that they can be reused. Figure 6.2 shows a diagram that presents how both dimensions can be explored in order to design the self-* properties under the self-* P2P approach.

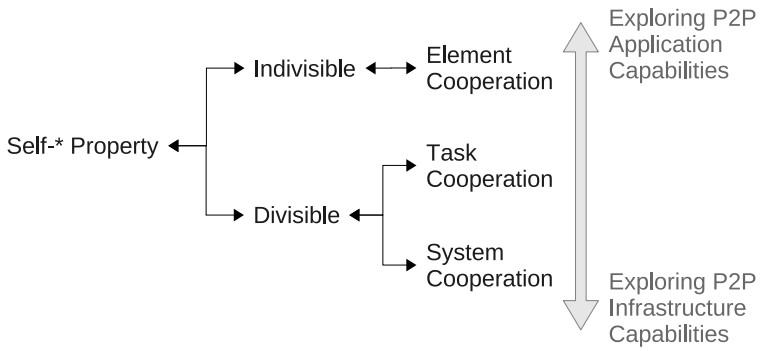


Figure 6.2: Relationship of self-* properties nature and cooperative degrees

The process of designing a self-* property can start with the definition of which kind of cooperative degree is meant to be achieved or which is the desired type of self-* property nature. Once one of these dimensions is fixed the second one is a consequence as it is illustrated in Figure 6.2. In addition, the definition of the design of the self-* property implicitly indicates which kind of P2P capabilities can be explored. It is important to remark that infrastructure P2P capabilities can always be explored by any design. However this tends not to be valid to P2P application capabilities.

The element cooperation enables the possibility of exploring P2P application capabilities by developing applications based on complementary perspectives and rendezvous-based decisions (when two neighbors agree on executing an action), and exploring parallel programming techniques (such as SPMD) to make neighbor peers to cooperate. In the opposite side, system cooperation tends to explore the P2P infrastructure capabilities, like location and connectivity, in order to coordinate the use of distinct self-elements to perform an action. Despite the fact that the design of task cooperative self-* properties does not count on complementary perspectives, such design is still related to a single self-* property. Thus, it is possible to embed P2P application capabilities on the task logic.

The cooperative degrees here presented are expected to be an start point for enhancing the process of changing the logic of network management applications from centralized and hierarchical to cooperative by exploring the P2P application capabilities. The experience achieved with this thesis highlights some P2P application capabilities suitable for network management solutions but does not exhaust the topic. Finally, the dimensions of the self-* P2P approach are envisioned to become a guideline on the design of truly distributed and cooperative network and services management solutions that aim at reducing the human intervention by endowing the systems with self-* capabilities.

6.4 Self-* P2P in the Network Management Scenario

It is a common practice in the network management community to import techniques and technologies from other research areas in order to solve its challenges. For example, over the years, the parallel and distributed computing area lent technologies such as CORBA, Web services, and P2P so that they could be explored and incorporated in the network management field. Multi-agent systems contributed with techniques for developing mobile code and intelligent agents for network management. Recently, techniques such as ontologies, bio-inspired computing, and autonomic computing have been borrowed from self-management research and employed on network management. This thesis touched these three research areas, borrowing distinct concepts and techniques and applying them to design fully distributed and cooperative network management applications. In fact, the self-* P2P approach uses the intersections of these three areas of research, and the placement of the developed approach is illustrated in Figure 6.3.

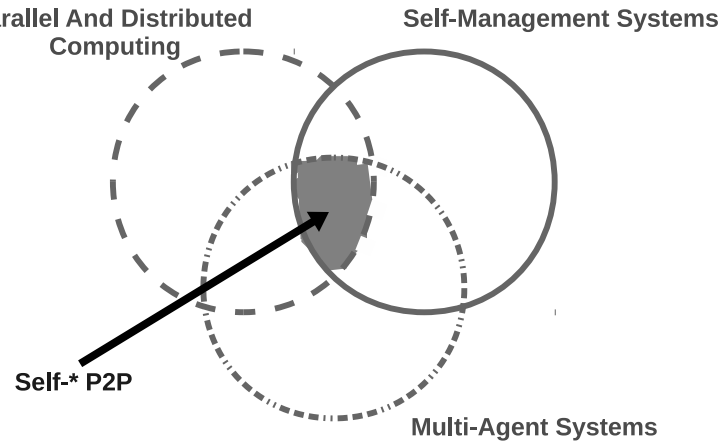


Figure 6.3: Diagram of self-* P2P placement among techniques and technologies employed on network management

The exact intersection among parallel and distributed computing, self-management systems, and multi-agent systems gathers the features of those areas of research that were combined by this thesis in order to design the self-* P2P approach for network management. From parallel and distributed computing, the proposed approach uses the P2P infrastructure capabilities, the attributes of concurrent models, and parallel/distributed techniques for the development of applications. Self-management systems contributed with the idea of embedding inside the mangers of the system the management of tasks that can be removed from the human administrator scope. Finally, imported from the multi-agent system, the sense of cooperative behavior was simplified to coupe with the network management requirements (without introducing many issues specific from the multi-agents (FERBER, 1999)) and strongly used.

Most of the times, it is not an easy task to draw the big picture of the relationship between the techniques and technologies borrowed from other areas used on network management because there are very tenuous borders among them. Therefore, the diagram presented in Figure 6.3 draws the intersections already investigated by the network management community and reviewed in Chapter 2, and also defines the placement of the self-* P2P approach investigated in this thesis.

6.5 Summary

This chapter describes in a first moment the analysis of the self-* P2P approach in the light of the Parallel and Distributed Computing research area. The parallel processing paradigms are strongly related to classes of algorithms that can be developed for each paradigm. This way, transposing these parallel processing paradigms helps on identifying the classes of applications developed for the common network management approaches and what is the difference for the self-* P2P approach. In addition, this chapter also establishes a relationship between concurrency attributes and the integration issues in order to demonstrate the fully distributed, parallel, and cooperative features of the self-* P2P approach. In a second moment, the integration issues are analyzed in more details. A direct comparison between the designed solutions for each one of the case studies is analyzed. The conclusion of this analysis shows that a simplistic analysis of the features of the developed solutions based on the self-* P2P approach hides further important characteristics of this approach. These characteristics are, in fact, dimensions that define which are the intrinsic features of the solution to be designed. Two dimensions were identified: the self-* property nature (indivisible and divisible) and the cooperative degrees (element, task, and system). There is a direct relationship between these two dimension, so that once one is fixed the other dimension is also determined. Finally, the self-* P2P is presented in terms of its placement in the scope of network management scenario, once it is related to different techniques and areas that have been already used in this scenario.

7 CONCLUSIONS AND FUTURE WORK

The investigation carried on this thesis shows the process of merging the high level concepts of self-* properties with P2P in terms of its infrastructure and application capabilities. In order to drive the directions of this investigation, a hypothesis was defined following the belief that "*the combination of self-* properties and P2P techniques enables the design and development of truly distributed and cooperative network management applications*". The methodology used to verify the validation of such hypothesis was based on the definition of management requirements, integration issues for the design of the joint use of self-* properties and P2P, and the identification/development of case studies. In this thesis, two case studies were discussed. The first one explored a self-healing P2P design for reliability of network management platforms, while the second case study examined a self-organizing P2P solution for resource management (in special network traffic) on network environments with underlaying and overlay infrastructures.

Analyzing these case studies it is possible to conclude that the first case study had an importance in terms of breaking the traditional paradigms of using MbD on top of P2P infrastructures for network management. The design of the self-healing P2P solution embedded the healing task inside the logic of the peers. There is no hierarchy or centralized entities. The peers are in the same "plane" and intentionally cooperate to provide the self-* property. This way, the first case study opened the possibility of rethinking how self-* properties could be designed using the whole potential of cooperative behavior offered by P2P-based solutions. Embracing this break through on traditional network management approaches, the second case study - self-organizing P2P solution - gave a step further on the development of P2P application capabilities for the design of a self-* properties applied to network management. The exploitation of algorithms based on complementary perspectives and neighbor-based decision-making gave another perspective to network management, where no external entities are needed to manage the environment. The consequence of such design is a truly distributed and cooperative network management environment, where the presence of the human administrator is minimized and the managers are embedded inside the managed elements and not in the borders of the system.

In addition, the observation of the self-* P2P solutions proposed in the case studies showed that the design of divisible or indivisible self-* properties (*i.e.*, whether they are composed of one or more self-elements) plays a key role on the integration of self-* properties and P2P for the network management. When divisible self-* properties are considered, the designed solution restrains the employment of complementary perspectives because the logic of the self-* property is not enclosed in a single type of self-element but it is spread in different types of self-elements. The advantages of using divisible self-* properties approach are modularity and potential re-usability of self-elements, and the algorithms behind of the self-elements are easy to be designed. The risk of using

this approach lies on the tendency of exploring solely P2P infrastructure capabilities for designing the self-* property, and this can lead to the traditional use of MbD structures.

In contrast, the design of indivisible self-* properties using P2P application capabilities represents the possibility of defining a new paradigm on network management research because of the complementary perspectives embedded inside each self-element. For example, one might think that SNMP-based network management approach also present complementary perspectives, represented by the manager (that requests tasks) and the agent (that executes the orders). However, the difference of the complementary perspectives from SNMP-based and self-* P2P with indivisible self-* properties[1] lies on the fact that in the first case these perspectives are not part of a parallel/concurrent application but they are distributed applications that communicate, while on indivisible self-* P2P approach these perspectives are conceived as a parallel/concurrent and distributed application. Despite the advantages introduced by this approach (*i.e.*, truly distributed and cooperative network management solutions) its major drawback is the complexity of identifying and developing the complementary perspectives of the same problem. Nevertheless, the same way that efforts were expended on the definition of MIBs and SNMP protocol it is possible to think on building information models to represent the complementary perspectives and also to develop frameworks to support the development of parallel/concurrent applications for network management based on the ideas introduced by the indivisible self-* P2P approach. To conclude, the answers for the fundamental questions, the contributions of this thesis, and the future work are described as follows.

7.1 Answers for the Fundamental Questions

The intention behind the definition of the fundamental questions was to help on raising the main points to be analyzed during the investigation of the hypothesis and to establish the way to reach the contributions of this thesis. Therefore, the description below summarizes and highlights the major characteristic of the answers of each fundamental question.

Fundamental Question and Contribution I. *What are the characteristics introduced by the self-* properties and P2P techniques on the design and execution of network management solutions?*

Answer. The first remarkable feature is that the self-* P2P approach changes the angle that network management solutions are typically developed. Instead of focusing on the morphological aspects of the solution (such as APIs, protocols, architectures, and frameworks) the main concerns are related to the design of the management algorithms. The focus is to explore the parallel and cooperative behavior of the management peers running the management algorithms. The second important characteristic is the possibility of embedding the management inside the managed elements without the need of external managers. Finally, due to the fact that the management is embedded inside the network itself, there is a better support for simultaneous and parallel execution of management actions, once peers in different parts of the network can take decisions based on their information and from their neighbors without waiting for higher level managers.

[1] It is not argued here the obvious difference that every SNMP-based solution is extremely dependent on human intervention and that the self-* P2P one minimizes such intervention.

Fundamental Question and Contribution II. *What are the benefits of employing self-* P2P techniques on building network management solutions in comparison to self-* centralized or self-* hierarchical approaches?*

Answer. The major benefit is the possibility to reach a fully distributed and true cooperative behavior in the network management solution. A self-* property by itself is supposed to reduce the human intervention and can be designed in different manners. The centralized and hierarchical approaches suffer from well known problems (*e.g.*, being more susceptible to failures, lack of scalability, bottlenecks for traffic and processing tasks), meanwhile the P2P approach enables the use of P2P infrastructures (like connectivity and location) and the cooperative aspect of applications associated with this approach.

Fundamental Question and Contribution III. *Which kind of costs do the self-* P2P-based network management approach impose?*

Answer. The relevant cost that is introduced by the self-* P2P approach is associated with the complexity of the design of the solution. The development of a network management solution under this approach has to take into account the following phases of design: the mapping process of a self-* property into a unique self-element or a set of them, which also implies the definition of which cooperative degree is chosen; the incorporation of the integration issues into the solution; the development of parallel/concurrent algorithms able to execute the task behind the self-* property using the neighbor-based decision-making. These development costs can be "paid" in the case of very complex, large, and dynamic networks, but in smaller scenarios it might not be the best solution.

Finally, the characteristics of the self-* P2P approach lead to a new type of network management applications. In this new type of applications, the problems of the network are handled by the services and devices of the network themselves in a fully distributed and cooperative fashion.

7.2 Contributions

The contributions of this thesis can be divided into: conceptual and punctual ones. Conceptual contributions could be identified due to the investigations of the literature and the experiences gathered during the development of the case studies. In contrast, punctual contributions are associated with individual solutions developed for each case study analyzed in this thesis. Below, both contributions are listed.

- This thesis present a survey relating the three topics: autonomic computing and self-* properties, P2P, and network and service management. In fact, this survey helps to organize the works proposed in the literature in face of the relationship of these three areas.

- The self-* P2P approach changes the angle that network management solutions are typically developed from morphological aspects (such as APIs, protocols, architectures, and frameworks) to the design of the management algorithms. The focus is to explore the parallel and cooperative behavior of the management peers running the management algorithms.

- The self-* P2P approach proposes techniques (based on the case studies) for enabling a fully distributed and true cooperative behavior in the network management solution.

- This thesis presents a suitable approach to embedded the management inside the managed elements without the need of external managers, in its turn, enabling a self-management behavior. The consequence of the embodiment of management actions is a better support for designing and developing simultaneous and parallel execution of management actions.

- The work developed in this thesis shapes the borders and intersection among parallel and distributed computing, self-management, and multi-agent systems applied to network management.

- Punctual contributions of the self-healing P2P solution

  – Definition of a management platform able to heal failed management instances without human intervention using a cooperative mechanism based on intra and inter group-communication to execute management actions without external management entities.

  – The evaluation of the designed solution showed the trade-off between the recovery time and associated network traffic. The determination of such trade-off is important because it shows when a faster recovery process consumes too much network bandwidth. In contrast, it also shows when excessively saved bandwidth leads to services that remain unavailable longer.

- Punctual contributions of the self-organizing P2P solution

  – Definition of a distributed management architecture for network virtualization. This architecture uses network management applications based on complementary perspectives and P2P interactions.

  – Introduction of P2P interactions embedded inside the network elements in order to solve traffic engineering problems. Generally, traffic engineering solutions use external entities (typically centralized or hierarchical) that monitor the network elements and try to solve the problem. The neighbor peer interactions introduced by the self-organizing P2P solution represent a break through on traffic engineering.

  – Definition of heuristics to identify cut-through traffic on substrate networks solely based on local information without using techniques of packet level inspection.

Summarizing, this thesis showed possibilities to rethink the way of designing and performing network management. The focus changes from morphological aspects (*i.e.*, protocols, frameworks, architectures) to algorithmic aspects of the management. Moreover, this thesis shows a clear alternative for structuring and really creating fully distributed and cooperative management network applications.

7.3 Future Work

The investigation developed on this thesis leads to the identification of further opportunities of research. These opportunities are described in this section as future work. First, the future work related to the design of self-* P2P solutions is presented. In a second moment, the text depicts the open issues associated with the specific solutions developed for each one of the case studies, and those open issues also constitute future work opportunities.

Multiples Self-* Properties Together. One interesting opportunity of research is to understand which are the consequences of designing different self-* properties using the self-* P2P approach. The interactions of the self-elements associated with each self-* P2P solution can be designed either to cooperate among all of them (system cooperative degree) or to ignore the existence of other self-elements of distinct self-* properties. These interactions could lead to the design of coupled and decoupled self-* properties, and these designs could enclose trade-offs between the transparent and intentional cooperative posture.

Formalization of Self-* P2P Approach. It is interesting to think in manners to formalize the concepts elucidated with the design of self-* P2P solutions for distributed and cooperative network management application, so that this formal method can be applied for different types of self-* properties. Different from information models, application models for network management solutions are not very common, and could constitute a good research opportunity and a manner to improve the methodology and knowledge of the area.

Open Issues of Self-healing P2P Solution. The case study associated with the self-healing P2P approach has some punctual open issues that could be investigated as future work. Below, these issues are presented.

- Optimize the detection mechanism because the current version is responsible for a considerable amount of the generated traffic.
- Investigate how service policies may impact the consumed network bandwidth and recovery time of the self-healing P2P approach here proposed.
- Verify what are the impacts on traffic consumption and response time when designing the self-healing and self-configuration as a single self-element.
- Improve the recovering process of the management services. Instead of choosing random peers to deploy new instances of the failed management services, the proximity of the monitored entities should be considered.

Open Issues of Self-organizing P2P Solution. The work developed on the case study associated with the self-organizing P2P also presented some punctual issues that could be investigated as future work. Those issues are listed below.

- Determine if in a long term the decisions taken by pairs of peers running the self-organizing control loop are being profitable for the entire substrate network. In this sense, it is interesting to investigate self-optimization strategies for the self-organizing P2P solution.
- Verify what are the drawbacks and benefits of cross-layer interaction between underlay and overlay infrastructures.

- Investigate an economical model to determine the number of virtual pipes and nodes associated to a virtual network.
- Identify the trade-off between costs of buying slices of resources versus the flexibility on managing the substrate network resources.

The list of future work presented above represents the major opportunities of research that can be directly derived from the work presented in this thesis. Nevertheless, there are other opportunities that can be explored under a self-* P2P context, such as providing security mechanism for the execution of self-* and fully distributed and cooperative management actions, or exploring cross-layer techniques to enhance self-* P2P solutions.

REFERENCES

4WARD. **The FP7 4WARD Project**. Available at `http://www.4ward-project.eu/`. Accessed in March 2010.

AGARWAL, M. et al. AutoMate: enabling autonomic applications on the grid. In: AUTONOMIC COMPUTING WORKSHOP, 2003. **Proceedings...** [S.l.: s.n.], 2003. p.48 – 57.

AGHAZADEHY, Z. et al. PeerStar: an attractive alternative to existing peer-to-peer topologies. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, 2009. ISCC 2009. **Proceedings...** [S.l.: s.n.], 2009. p.128–134.

AGRAWAL, D. et al. Planning and Managing the IPTV Service Deployment. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT (IM 2007), 10. **Proceedings...** [S.l.: s.n.], 2007. p.353 – 362.

AKASHI, O. et al. Detection and Diagnosis of Inter-AS Routing Anomalies by Cooperative Intelligent Agents. In: IFIP/IEEE INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS: OPERATIONS AND MANAGEMENT, DSOM 2006, 16., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2005. p.181–192. (Lecture Notes in Computer Science, v.3775).

AL-OQILY, I.; KARMOUCH, A. A self-organizing composition towards autonomic overlay networks. In: IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 2008. NOMS 2008. **Proceedings...** [S.l.: s.n.], 2008. p.287 – 294.

ALHAISONI, M.; LIOTTA, A. Characterization of signaling and traffic in Joost. **Peer-to-Peer Networking and Applications**, [S.l.], v.2, n.1, p.75–83, 2009.

ALTMANN, J.; BEDANE, Z. A P2P File Sharing Network Topology Formation Algorithm Based on Social Network Information. In: IEEE INFOCOM WORKSHOPS 2009. **Proceedings...** [S.l.: s.n.], 2009. p.1 – 6.

ALVES, R. S. et al. A Protocol for Atomic Deployment of Management Policies in QoS-Enabled Networks. In: IEEE INTERNATIONAL WORKSHOP ON IP OPERATIONS AND MANAGEMENT - AUTONOMIC PRINCIPLES OF IP OPERATIONS AND MANAGEMENT, IPOM 2006, 6., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2006. p.132–143. (Lecture Notes in Computer Science, v.4268).

ANA. **Autonomic Network Architecture**. Available at `http://www.ana-project.org/web/`. Accessed in March 2010.

ANALOUI, M.; JAMALI, S. Congestion Control in the Internet: inspiration from balanced food chains in the nature. **Journal of Network and Systems Management**, New York, NY, USA, v.16, n.1, p.1–10, 2008.

ANDROUTSELLIS-THEOTOKIS, S.; SPINELLIS, D. A survey of peer-to-peer content distribution technologies. **ACM Computing Surveys**, New York, NY, USA, v.36, n.4, p.335–371, 2004.

ANEROUSIS, N. An Architecture for Building Scalable, Web-Based Management Services. **Journal of Network and Systems Management**, [S.l.], v.7, n.1, p.73–104, March 1999.

ANTHONY, R.; BUTLER, A.; IBRAHIM, M. Layered Autonomic Systems. In: SECOND INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING, 2005. ICAC 2005. **Proceedings...** [S.l.: s.n.], 2005. p.383 – 384.

APPLEBY, K. et al. Oceano-SLA based management of a computing utility. In: IEEE/IFIP INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.855 – 868.

AWERBUCH, B.; KHANDEKAR, R. Distributed network monitoring and multicommodity flows: a primal-dual approach. In: TWENTY-SIXTH ANNUAL ACM SYMPOSIUM ON PRINCIPLES OF DISTRIBUTED COMPUTING, PODC '07, New York, NY, USA. **Proceedings...** ACM, 2007. p.284–291.

AYSAL, T.; BARNER, K. On the Convergence of Perturbed Non-Stationary Consensus Algorithms. In: IEEE INFOCOM 2009. THE 28TH CONFERENCE ON COMPUTER COMMUNICATIONS. **Proceedings...** [S.l.: s.n.], 2009. p.2132 – 2140.

BADONNEL, R.; STATE, R.; FESTOR, O. A Probabilistic Approach for Managing Mobile Ad-Hoc Networks. **IEEE Transactions on Network and Service Management**, [S.l.], v.4, n.1, p.39–50, June 2007.

BALASUBRAMANIAM, S. et al. Biologically Inspired Self-Governance and Self-Organisation for Autonomic Networks. In: BIO-INSPIRED MODELS OF NETWORK, INFORMATION AND COMPUTING SYSTEMS, 2006., 1. **Proceedings...** [S.l.: s.n.], 2006. p.1 – 7.

BALDI, M. Triple Play Support for the Next Generation Internet. In: INTERNATIONAL TELECOMMUNICATIONS NETWORK STRATEGY AND PLANNING SYMPOSIUM, 2006. NETWORKS 2006., 12. **Proceedings...** [S.l.: s.n.], 2006. p.1 – 7.

BALIOSIAN, J. et al. Policy-based self-healing for radio access networks. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM: PERVASIVE MANAGEMENT FOR UBIOQUITOUS NETWORKS AND SERVICES, NOMS 2008. **Proceedings...** [S.l.: s.n.], 2008. p.1007–1010.

BANAEI-KASHANI, F.; SHAHABIA, C. Partial read from peer-to-peer databases. **Computer Communications - Special Issue: Foundation of Peer-to-Peer Computing**, [S.l.], v.31, n.2, p.332–345, February 2008.

BARBERA, M. et al. CLAPS: a cross-layer analysis platform for p2p video streaming. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 2007. ICC '07. **Proceedings...** [S.l.: s.n.], 2007. p.50 – 56.

BARILLAUD, F.; DERI, L.; FERIDUN, M. Network Management Using Internet Technologies. In: FIFTH IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT. **Proceedings...** [S.l.: s.n.], 1997. p.61–70.

BARSHAN, M.; FATHY, M.; YOUSEFI, S. Fault-Tolerant Architecture for Peer to Peer Network Management Systems. In: THE 9TH INTERNATIONAL CONFERENCE ON NEXT GENERATION WIRED/WIRELESS NETWORKING, NEW2AN 2009 AND SECOND CONFERENCE ON SMART SPACES, RUSMART 2009, Heidelberg, Germany. **Proceedings...** Springer Berlin, 2009. p.241–252. (Lecture Notes in Computer Science, v.5764).

BAUDE, F.; HENRIO, L.; NAOUMENKO, P. A component platform for experimenting with autonomic composition. In: INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING AND COMMUNICATION SYSTEMS, AUTONOMICS '07., 1., ICST, Brussels, Belgium, Belgium. **Proceedings...** ICST (Institute for Computer Sciences: Social-Informatics and Telecommunications Engineering), 2007. p.1–9.

BEJAN, A.; GHOSH, S. Self-optimizing DHTs Using Request Profiling. In: INTERNATIONAL CONFERENCE ON PRINCIPLES OF DISTRIBUTED SYSTEMS, OPODIS 2004, 8., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2004. p.140–153. (Lecture Notes in Computer Science, v.3544).

BENEDIKTSSON, J.; SWAIN, P. Consensus theoretic classification methods. **IEEE Transactions on Systems, Man and Cybernetics**, [S.l.], v.22, n.4, p.688 – 704, July-Aug. 1992.

BERGLUND, A. et al. Toward Goal-Based Autonomic Networking. In: IEEE GLOBECOM WORKSHOPS, 2008. **Proceedings...** [S.l.: s.n.], 2008. p.1 – 5.

BERL, A. et al. Management of Virtual Networks. In: PROCEEDINGS OF FOURTH IEEE/IFIP INTERNATIONAL WORKSHOP ON END-TO-END VIRTUALIZATION AND GRID MANAGEMENT, EVGM 2008. **Proceedings...** [S.l.: s.n.], 2008. p.197 – 202.

BERNS, A.; GHOSH, S. Dissecting Self-* Properties. In: THIRD IEEE INTERNATIONAL CONFERENCE ON SELF-ADAPTIVE AND SELF-ORGANIZING SYSTEMS, 2009. SASO '09. **Proceedings...** [S.l.: s.n.], 2009. p.10 – 19.

BEZERRA, R. S. et al. Um Sistema de Gerenciamento de Redes Baseado em Mashups. In: SBRC 2009 - 27O SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUIDOS. 2009. **Proceedings...** [S.l.: s.n.], 2009.

BEZERRA, R. S. et al. On the Feasibility of Web 2.0 Technologies for Network Management: a mashup-based approach. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM. NOMS 2010., 12. **Proceedings...** [S.l.: s.n.], 2010.

BHUSHAN, S.; SCHÖNWÄLDER, H. M. T. J. NCClient: a python library for netconf client applications. In: IEEE INTERNATIONAL WORKSHOP ON IP OPERATIONS AND MANAGEMENT, IPOM 2009, 9., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2009. p.143–154. (Lecture Notes in Computer Science, v.5843).

BINZENHÖFER, A. et al. A P2P-Based Framework for Distributed Network Management. In: WIRELESS SYSTEMS AND NETWORK ARCHITECTURES IN NEXT GENERATION INTERNET, Heidelberg, Germany. **Proceedings...** Springer-Berlim, 2006. p.198–210. (Lecture Notes in Computer Science, v.3883).

BISKUPSKI, B.; DOWLING, J.; SACHA, J. Properties and mechanisms of self-organizing MANET and P2P systems. **ACM Transactions on Autonomous and Adaptive Systems (TAAS)**, New York, NY, USA, v.2, n.1, p.1, 2007.

BJUREFORS, F.; LARZON, L.; GOLD, R. Performance of Pastry in a heterogeneous system. In: FOURTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.278 – 279.

BOONMA, P.; SUZUKI, J. Exploring self-star properties in cognitive sensor networking. In: SPECTS 2008. INTERNATIONAL SYMPOSIUM ON PERFORMANCE EVALUATION OF COMPUTER AND TELECOMMUNICATION SYSTEMS, 2008., Edinburgh. **Proceedings...** [S.l.: s.n.], 2008. p.36 – 43.

BOUABENE, G. et al. The autonomic network architecture (ANA). **IEEE Journal on Selected Areas in Communications**, New York, USA, v.28, n.1, p.4 – 14, January 2010.

BRUNNER, M. et al. Towards Ambient Networks Management. In: MOBILITY AWARE TECHNOLOGIES AND APPLICATIONS. **Proceedings...** Heidelberg: Springer-Berlim, 2005. p.215–229. (Lecture Notes in Computer Science, v.3744).

CALLAWAY, R. et al. An Autonomic Service Delivery Platform for Service-Oriented Network Environments. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 2008. ICC '08. **Proceedings...** [S.l.: s.n.], 2008. p.327 – 331.

CAMPOS, R.; RICARDO, M. Dynamic and automatic connection of personal area networks to the global internet. In: IWCMC '06: PROCEEDINGS OF THE 2006 INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS AND MOBILE COMPUTING, New York, NY, USA. **Proceedings...** ACM, 2006. p.581–586.

CAPONE, A.; ELIAS, J.; MARTIGNON, F. Models and Algorithms for the Design of Service Overlay Networks. **IEEE Transactions on Network and Service Management**, [S.l.], v.5, n.3, p.143–156, September 2008.

CARRERAS, I. et al. BIONETS: bio-inspired networking for pervasive communication environments. **IEEE Transactions on Vehicular Technology**, [S.l.], v.56, n.1, p.218 – 229, January 2007.

CASTELLI, G.; MENEZES, R.; ZAMBONELLI, F. Self-organized control of knowledge generation in pervasive computing systems. In: SAC '09: PROCEEDINGS OF THE 2009 ACM SYMPOSIUM ON APPLIED COMPUTING, New York, NY, USA. **Proceedings...** ACM, 2009. p.1202–1208.

CAVENDISH, D. Operation, administration, and maintenance of Ethernet services in wide area networks. **IEEE Communications Magazine**, [S.l.], v.42, n.3, p.72 – 79, March 2004.

CHAKRAVARTI, A.; BAUMGARTNER, G.; LAURIA, M. The organic grid: self-organizing computation on a peer-to-peer network. **IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans**, [S.l.], v.35, n.3, p.373 – 384, May 2005.

CHAPARADZA, R.; COSKUN, H.; SCHIEFERDECKER, I. Addressing some challenges in autonomic monitoring in self-managing networks. In: IEEE INTERNATIONAL CONFERENCE ON NETWORKS, 2005., 13. **Proceedings...** [S.l.: s.n.], 2005. v.2, p.6. CDROM.

CHAPARADZA, R.; COSKUN, H.; SCHIEFERDECKER, I. Addressing some challenges in autonomic monitoring in self-managing networks. In: IEEE INTERNATIONAL CONFERENCE ON NETWORKS, 2005. JOINTLY HELD WITH THE 2005 IEEE 7TH MALAYSIA INTERNATIONAL CONFERENCE ON COMMUNICATION, 13. **Proceedings...** [S.l.: s.n.], 2005. v.2, p.6.

CHAUDHURI, S.; NARASAYYA, V. Self-tuning database systems: a decade of progress. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB '07, 33. **Proceedings...** VLDB Endowment, 2007. p.3–14.

CHEN, S.; WANG, X.; JAJODIA, S. On the anonymity and traceability of peer-to-peer VoIP calls. **IEEE Network**, [S.l.], v.20, n.5, p.32 – 37, September-October 2006.

CHEN, Y.-F. et al. VP2P: a virtual machine-based p2p testbed for vod delivery. In: IEEE CONSUMER COMMUNICATIONS AND NETWORKING CONFERENCE, 2009. CCNC 2009., 6. **Proceedings...** [S.l.: s.n.], 2009. p.1–5.

CHENG, B.-N.; YUKSEL, M.; KALYANARAMAN, S. Virtual Direction Routing for overlay networks. In: IEEE NINTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2009. P2P '09. **Proceedings...** [S.l.: s.n.], 2009. p.61–70.

CHERKAOUI, O. et al. QOS metrics tool using management by delegation. In: IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 1998. NOMS 98. **Proceedings...** [S.l.: s.n.], 1998. v.3, p.836 – 839.

CHIANG, F.; BRAUN, R.; AGBINYA, J. I. Self-Configuration of Network Services with Biologically Inspired Learning and Adaptation. **Journal of Network and Systems Management**, New York, NY, USA, v.15, n.1, p.87–116, 2007.

CHONG, M. Y. et al. Goal-Based Service Creation Using Autonomic Entities. In: FOURTH IEEE INTERNATIONAL WORKSHOP ON MODELLING AUTONOMIC COMMUNICATIONS ENVIRONMENTS, MACE 2009, Heidelberg, Germany. **Proceedings...** Springer Berlin, 2009. p.29–43. (Lecture Notes in Computer Science, v.5844).

CHOURMOUZIADIS, A.; DUQUE, O. G.; PAVLOU, G. Experiences in using MUWS for scalable distributed monitoring. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2009. IM '09. **Proceedings...** [S.l.: s.n.], 2009. p.561 – 568.

CHOWDHURY, N. M. M. K.; BOUTABA, R. Network virtualization: state of the art and research challenges. **IEEE Communications Magazine**, [S.l.], v.47, n.7, p.20 – 26, July 2009.

CHOWDHURY, N.; RAHMAN, M.; BOUTABA, R. Virtual Network Embedding with Coordinated Node and Link Mapping. In: THE 28TH CONFERENCE ON COMPUTER COMMUNICATIONS. IEEE INFOCOM 2009. **Proceedings...** [S.l.: s.n.], 2009. p.783 – 791.

COELHO, J.; GASPARY, L.; TAROUCO, L. How much management is management enough? Providing monitoring processes with online adaptation and learning capability. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2009. IM '09. **Proceedings...** [S.l.: s.n.], 2009. p.299 – 302.

CORPORATION, Q. **QNEXT - Mobile/PC Unified Communications and Sharing Suite**. Available at `http://www.qnext.com/`. Accessed in March 2010.

CUI, D.; GUTIÉRREZ, J. A. An integrated network management framework using CORBA, mobile agents and web-based technologies. **Information management: support systems & multimedia technology**, Hershey, PA, USA, p.298–309, 2003.

CURBERA, F. et al. The next step in Web services. **Communications of the ACM**, New York, NY, USA, v.46, n.10, p.29–34, 2003.

DEGRANDE, N. et al. Increasing the User Perceived Quality for IPTV Services. **IEEE Communications Magazine**, [S.l.], v.46, n.2, p.94–99, February 2008.

DEMESTICHAS, K. et al. Towards Cognitive B3G Networks: autonomic management of access points. In: IST MOBILE AND WIRELESS COMMUNICATIONS SUMMIT, 2007., 16., Budapest. **Proceedings...** [S.l.: s.n.], 2007. p.1–5.

DHIWAL, A. et al. MailZoro - email based P2P file sharing. In: IEEE INTERNATIONAL CONFERENCE ON NETWORKS, 2008. ICON 2008., 16. **Proceedings...** [S.l.: s.n.], 2008. p.1–5.

DHURANDHER, S. et al. A Swarm Intelligence-based P2P file sharing protocol using Bee Algorithm. In: IEEE/ACS INTERNATIONAL CONFERENCE ON COMPUTER SYSTEMS AND APPLICATIONS, 2009. AICCSA 2009. **Proceedings...** [S.l.: s.n.], 2009. p.690 – 696.

DIAO, Y. et al. A control theory foundation for self-managing computing systems. **IEEE Journal on Selected Areas in Communications**, [S.l.], v.23, n.12, p.2213 – 2222, December 2005.

DING, X. et al. A Failure Detection Model Based on Message Delay Prediction. In: EIGHTH INTERNATIONAL CONFERENCE ON GRID AND COOPERATIVE COMPUTING, 2009. GCC '09. **Proceedings...** [S.l.: s.n.], 2009. p.24 – 30.

DOBSON, S. et al. Fulfilling the Vision of Autonomic Computing. **IEEE Computer**, [S.l.], v.43, n.1, p.35 – 41, January 2010.

DONG, L.; YUE-LONG, Z. Distributed Relational Data Sharing Based on P2P. In: INTERNATIONAL CONFERENCE ON NEW TRENDS IN INFORMATION AND SERVICE SCIENCE, 2009. NISS '09. **Proceedings...** [S.l.: s.n.], 2009. p.378 – 383.

DONG, X. et al. Autonomia: an autonomic computing environment. In: IEEE INTERNATIONAL PERFORMANCE, COMPUTING, AND COMMUNICATIONS CONFERENCE, 2003, 2003., Phoenix, Arizona, US. **Proceedings...** [S.l.: s.n.], 2003. p.61–68.

DU, T. C.; LI, E. Y.; CHANG, A.-P. Mobile agents in distributed network management. **Communications of the ACM**, New York, NY, USA, v.46, n.7, p.127–132, July 2003.

DUDKOWSKI, D. et al. Architectural principles and elements of in-network management. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2009. IM '09. **Proceedings...** [S.l.: s.n.], 2009. p.529 – 536.

EFIPSANS. **EFIPSANS Project - Exposing the Features in IP version Six protocols that can be exploited/extended for the purposes of designing/building Autonomic Networks and Services**. Available at `http://www.efipsans.org/index.php`. Accessed in March 2010.

EGI, N. et al. Evaluating Xen for Router Virtualization. In: PROCEEDINGS OF 16TH INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS AND NETWORKS, 2007. ICCCN 2007. **Proceedings...** [S.l.: s.n.], 2007. p.1256–1261.

EINSTEIN@HOME. Available at `http://einstein.phys.uwm.edu/`. Accessed in March 2010.

ELLIOTT, C.; FALK, A. An update on the GENI project. **SIGCOMM Computer Communication Review**, New York, NY, USA, v.39, n.3, p.28–34, 2009.

FALLON, L. et al. Self-forming Network Management Topologies in the Madeira Management System. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS INFRASTRUCTURE, MANAGEMENT AND SECURITY (AIMS'07), 1., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2007. p.61–72. (Lecture Notes in Computer Science, v.4543).

FAN, B.; LUI, J.; CHIU, D.-M. The Design Trade-Offs of BitTorrent-Like File Sharing Protocols. **IEEE/ACM Transactions on Networking**, [S.l.], v.17, n.2, p.365 – 376, April 2009.

FARHA, R.; LEON-GARCIA, A. Operations Research Methods for Autonomic Resource Management. **Journal of Network and Systems Management**, [S.l.], v.17, n.1-2, p.157–182, June 2009.

FEAMSTER, N.; GAO, L.; REXFORD, J. How to lease the Internet in your spare time. **ACM SIGCOMM Computer Communications Review**, [S.l.], 2007.

FERBER, J. (Ed.). **Multi-Agent System**: an introduction to distributed artificial intelligence. London, UK: Addison Wesley, 1999.

FIORESE, A.; SIMÕES, P.; BOAVIDA, F. A P2P-Based Approach to Cross-Domain Network and Service Management. In: THIRD INTERNATIONAL CONFERENCE ON

AUTONOMOUS INFRASTRUCTURES, MANAGEMENT AND SECURITY, AIMS 2009., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2009. p.179–182. (Lecture Notes in Computer Science, v.5637).

FIOREZE, T. et al. Comparing Web services with SNMP in a management by delegation environment. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2005. IM 2005., 9. **Proceedings...** [S.l.: s.n.], 2005. p.601 – 614.

FIOREZE, T. et al. A Policy-Based Hierarchical Approach for Management of Grids and Networks. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 2006. NOMS 2006., 10. **Proceedings...** [S.l.: s.n.], 2006. p.1 – 14.

FOLEY, C. et al. A Framework for In-Network Management in Heterogeneous Future Communication Networks. In: THIRD IEEE INTERNATIONAL WORKSHOP ON MODELLING AUTONOMIC COMMUNICATIONS ENVIRONMENTS, MACE 2008, Heidelberg, Germany. **Proceedings...** Springer Berlin, 2008. p.14–25. (Lecture Notes in Computer Science, v.5276).

FOUNDATION, A. S. **Welcome to Apache Axis2/Java**. Available at `http://ws.apache.org/axis2/`. Accessed in March 2010.

FRANZKE, M. et al. **Autonomic Internet Initial Framework - Deliverable D6.1**. Available at `http://www.4ward-project.eu/index.php?s=Deliverables`. Accessed in March 2010.

FUENTES, J. M.; VERGARA, J. E. L. de; CASTELLS, P. An Ontology-Based Approach to the Description and Execution of Composite Network Management Processes for Network Monitoring. In: IFIP/IEEE INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS: OPERATIONS AND MANAGEMENT, DSOM 2006, 17., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2006. p.86–97. (Lecture Notes in Computer Science, v.4269).

GALIS, A. et al. **Autonomic Internet Initial Framework - Deliverable D6.1**. Available at `http://ist-autoi.eu/autoi/index.php\#`. Accessed in March 2010.

GALIS, A. et al. Management Architecture and Systems for Future Internet Networks. **Towards the Future Internet - A European Research Perspective**, Amsterdan, Netherlands, v.0, p.112 – 122, 2009.

GANEK, A. G.; CORBI, T. The Dawning of the autonomic computing era. **IBM Systems Journal**, [S.l.], v.42, n.1, p.5–18, 2003.

GAO, L.; ZENG, G. An Innovative Hybrid P2P Location Network for P2P Workflow Systems. In: INTERNATIONAL CONFERENCE ON COMMUNICATION SOFTWARE AND NETWORKS, 2009. ICCSN '09. **Proceedings...** [S.l.: s.n.], 2009. p.61 – 65.

GELENBE, E.; LIU, P.; LAINE, J. Genetic Algorithms for Autonomic Route Discovery. In: IEEE WORKSHOP ON DISTRIBUTED INTELLIGENT SYSTEMS: COLLECTIVE INTELLIGENCE AND ITS APPLICATIONS, 2006. DIS 2006. **Proceedings...** [S.l.: s.n.], 2006. p.371 – 376.

GHAZALT, S. et al. Applying a self-configuring admission control algorithm in a new QoS architecture for IEEE 802.16 networks. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS. **Proceedings...** [S.l.: s.n.], 2008. p.1023–1028.

GOLDSZMIDT, G.; YEMINI, Y. Distributed management by delegation. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 1995, 15. **Proceedings...** [S.l.: s.n.], 1995. p.333–340.

GONG, L. JXTA: a network programming environment. **IEEE Internet Computing**, [S.l.], v.5, n.3, p.88 – 95, May-June 2001.

GONG, L. JXTA: a network programming environment. **IEEE Communications Magazine**, [S.l.], v.5, n.3, p.88–95, May 2005.

GPPD. **Parallel and Distributed Processing Group – GPPD**. Available in `http://gppd.inf.ufrgs.br/new/`. Accessed in March 2010.

GRANVILLE, L. Z. et al. Managing Computer Networks Using Peer-to-Peer Technologies. **IEEE Communications Magazine**, [S.l.], v.43, n.10, p.62–68, 2005.

GREENWOOD, D. Goal-Oriented Autonomic Business Process Modeling and Execution: engineering change management demonstration. In: INTERNATIONAL CONFERENCE ON BUSINESS PROCESS MANAGEMENT, BPM 2008., 6., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2008. p.390–393. (Lecture Notes in Computer Science, v.5240).

GUARDALBEN, L. et al. A cooperative Hide and Seek discovery over In Network Management. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM WORKSHOPS (NOMS WKSPS), 2010. **Proceedings...** [S.l.: s.n.], 2010. p.217 – 224.

GUDEMANN, M. et al. A Specification and Construction Paradigm for Organic Computing Systems. In: SECOND IEEE INTERNATIONAL CONFERENCE ON SELF-ADAPTIVE AND SELF-ORGANIZING SYSTEMS, 2008. SASO '08. **Proceedings...** [S.l.: s.n.], 2008. p.233 – 242.

GUNKEL, M. et al. Aggregation networks: cost comparison of wdm ring vs. double star topology. In: INTERNATIONAL CONFERENCE ON OPTICAL NETWORK DESIGN AND MODELING, 2008. ONDM 2008. **Proceedings...** [S.l.: s.n.], 2008. p.1 – 5.

GUO, F.; ZENG, B.; CUI, L. A Distributed Network Management Framework Based on Mobile Agents. In: THIRD INTERNATIONAL CONFERENCE ON MULTIMEDIA AND UBIQUITOUS ENGINEERING, 2009. MUE '09. **Proceedings...** [S.l.: s.n.], 2009. p.511 – 515.

GURGUIS, S. A.; ZEID, A. Towards autonomic web services: achieving self-healing using web services. In: WORKSHOP ON DESIGN AND EVOLUTION OF AUTONOMIC APPLICATION SOFTWARE, DEAS 2005, 2005., New York, NY, USA. **Proceedings...** ACM Press, 2005. p.1–5.

HAAS, J. J.; HU, Y.-C.; LABERTEAUX, P. K. In: VANET '09: PROCEEDINGS OF THE SIXTH ACM INTERNATIONAL WORKSHOP ON VEHICULAR INTERNETWORKING, New York, NY, USA. **Proceedings...** ACM, 2009. p.89–98.

HADJIANTONIS, A. M.; PAVLOU, G. Policy-based self-management of hybrid ad hoc networks for dynamic channel configuration. In: NOMS 2008. IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 2008., Salvador, Bahia. **Proceedings...** [S.l.: s.n.], 2008. p.433 – 440.

HAN, S.; LISLE, S.; NEHIB, G. IPTV Transport Architecture Alternatives and Economic Considerations. **IEEE Communications Magazine**, [S.l.], v.46, n.2, p.70–77, February 2008.

HARIRI, S. et al. The Autonomic Computing Paradigm. **Cluster Computing**, [S.l.], v.9, n.1, p.5–17, January 2006.

HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. **An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks**. 2002.

HERMANN, K. Self-organizing Replica Placement - A case study on emergence. In: FIRST INTERNATIONAL CONFERENCE ON SELF-ADAPTIVE AND SELF-ORGANIZING SYSTEMS, 2007. SASO '07. **Proceedings...** [S.l.: s.n.], 2007. p.13 – 22.

HONG, J. et al. Towards Bio-Inspired Self-Organization in Sensor Networks: applying the ant colony algorithm. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS, 2008. AINA 2008., 22. **Proceedings...** [S.l.: s.n.], 2008. p.1054 – 1061.

HORN, P. **Autonomic Computing**: ibm's perspective on the state of information technology. Available at `http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf`. Accessed in March 2010.

HOUIDI, I.; LOUATI, W.; ZEGHLACHE, D. A Distributed and Autonomic Virtual Network Mapping Framework. In: PROCEEDINGS OF FOURTH IEEE/IFIP INTERNATIONAL CONFERENCE ON AUTONOMIC AND AUTONOMOUS SYSTEMS, 2008. ICAS 2008. **Proceedings...** [S.l.: s.n.], 2008. p.241–247.

HUEBSCHER, M. C.; MCCANN, J. A. A survey of autonomic computing - degrees, models, and applications. **ACM Computing Surveys**, New York, NY, USA, v.40, n.3, p.1–28, 2008.

INC., G. **Google Talk**. Available at `http://www.google.com/talk/`. Accessed in March 2010.

INC., P. **PPLive - The most popular net TV int the world**. Available at `http://www.pplive.com/en/index.html`. Accessed in March 2010.

IZUMI, T.; SAITOH, A.; MASUZAWA, T. Timed uniform consensus resilient to crash and timing faults. In: INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.243 – 252.

JELGER, C. et al. Basic Abstractions for an Autonomic Network Architecture. In: IEEE INTERNATIONAL SYMPOSIUM ON A WORLD OF WIRELESS, MOBILE AND MULTIMEDIA NETWORKS, 2007. WOWMOM 2007., Espoo, Finland. **Proceedings...** [S.l.: s.n.], 2007. p.1–6.

JENNINGS, B. et al. Towards autonomic management of communications networks. **IEEE Communications Magazine**, [S.l.], v.45, n.10, p.112–121, October 2007.

JIMENEZ, R.; OSMANI, F.; KNUTSSON, B. Connectivity properties of Mainline BitTorrent DHT nodes. In: IEEE NINTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2009. P2P '09. **Proceedings...** [S.l.: s.n.], 2009. p.262–270.

JONES, D. et al. Knowledge delivery mechanism for autonomic overlay network management. In: ICAC '09: PROCEEDINGS OF THE 6TH INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING, New York, NY, USA. **Proceedings...** ACM, 2009. p.49–50.

JUN, L. et al. A Novel Network Management Architecture for Self-organizing Network. In: INTERNATIONAL CONFERENCE ON NETWORKING, ARCHITECTURE, AND STORAGE, 2007. NAS 2007. **Proceedings...** [S.l.: s.n.], 2007. p.146 – 154.

JYOTHI, B. S.; DHARANIPRAGADA, J. SyMon: defending large structured p2p systems against sybil attack. In: IEEE NINTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2009. P2P '09. **Proceedings...** [S.l.: s.n.], 2009. p.21–30.

KABATEPE, M.; VASTOLA, K. S. The fair distributed queue (FDQ) protocol for high-speed metropolitan-area networks. **IEEE/ACM Transactions on Networking (TON)**, Piscataway, NJ, USA, v.4, n.3, p.331–339, June 1996.

KAISER, G. et al. Kinesthetics eXtreme: an external infrastructure for monitoring distributed legacy systems. In: AUTONOMIC COMPUTING WORKSHOP. FIFTH ANNUAL INTERNATIONAL WORKSHOP ON ACTIVE MIDDLEWARE SERVICES. AMS 2003. **Proceedings...** [S.l.: s.n.], 2003. p.22 – 30.

KAMIENSKI, C. et al. On the Use of Peer-to-Peer Architectures for the Management of Highly Dynamic Environments. In: FOURTH ANNUAL IEEE INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS (PERCOM WORKSHOPS 2006). **Proceedings...** IEEE Press, 2006. 1 CD-ROM.

KANG, H. et al. Why Kad lookup fails. In: IEEE NINTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2009. P2P '09. **Proceedings...** [S.l.: s.n.], 2009. p.121–130.

KAR, S.; MOURA, J. Distributed Consensus Algorithms in Sensor Networks With Imperfect Communication: link failures and channel noise. **IEEE Transactions on Signal Processing**, [S.l.], v.57, n.1, p.355 – 369, January 2009.

KARNIK, A.; KUMAR, A. Distributed optimal self-organization in ad hoc wireless sensor networks. **IEEE/ACM Transactions on Networking**, Piscataway, NJ, USA, v.15, n.5, p.1035–1045, 2007.

KAUNE, S. et al. Embracing the Peer Next Door: proximity in kademlia. In: EIGHTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING , 2008. P2P '08. **Proceedings...** [S.l.: s.n.], 2008. p.343–350.

KEENEY, J.; LEWIS, D.; SULLIVAN, D. O. Ontological Semantics for Distributing Contextual Knowledge in Highly Distributed Autonomic Systems. **Journal of Network and Systems Management**, New York, USA, v.15, n.1, p.75–86, March 2007.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. **IEEE Computer**, [S.l.], v.32, n.1, p.41–50, 2003.

KHALID, A. et al. Survey of Frameworks, Architectures and Techniques in Autonomic Computing. In: FIFTH INTERNATIONAL CONFERENCE ON AUTONOMIC AND AUTONOMOUS SYSTEMS, 2009. ICAS '09. **Proceedings...** [S.l.: s.n.], 2009. p.220 – 225.

KHAN, M.; AWAIS, M.; SHAMAIL, S. Self-Configuration in Autonomic Systems Using Clustered CBR Approach. In: INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING, 2008. ICAC '08. **Proceedings...** [S.l.: s.n.], 2008. p.211 – 212.

KIM, H. et al. An Autonomic Approach to Integrated HPC Grid and Cloud Usage. In: FIFTH IEEE INTERNATIONAL CONFERENCE ON E-SCIENCE, 2009. E-SCIENCE '09. **Proceedings...** [S.l.: s.n.], 2009. p.366 – 373.

KIRI, Y.; SUGANO, M.; MURATA, M. Self-Organized Data-Gathering Scheme for Multi-Sink Sensor Networks Inspired by Swarm Intelligence. In: SASO '07. FIRST INTERNATIONAL CONFERENCE ON SELF-ADAPTIVE AND SELF-ORGANIZING SYSTEMS, 2007., Cambridge, MA. **Proceedings...** [S.l.: s.n.], 2007. p.161 – 172.

KOCH, F. et al. Distributed Artificial Intelligence for Network Management Systems - New Approaches. In: FIRST INTERNATIONAL WORKSHOP ON SERVICE ASSURANCE WITH PARTIAL AND INTERMITTENT RESOURCES, SAPIR 2004, Heidelberg, Germany. **Proceedings...** Springer Berlin, 2004. p.135–145. (Lecture Notes in Computer Science, v.3126).

KORPELA, E. et al. SETIhome-massively distributed computing for SETI. **Computing in Science & Engineering**, [S.l.], v.3, n.1, p.78 – 83, January-February 2001.

KOUSARIDAS, A. et al. Future internet elements: cognition and self-management design issues. In: AUTONOMICS '08: PROCEEDINGS OF THE 2ND INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING AND COMMUNICATION SYSTEMS, ICST, Brussels, Belgium, Belgium. **Proceedings...** ICST (Institute for Computer Sciences: Social-Informatics and Telecommunications Engineering), 2008. p.1–6.

LABORATORY, B.; WASHINGTON, U. of; COMMONS, R. **Rosetta@Home**. Available at `http://boinc.bakerlab.org/rosetta/`. Accessed in March 2010.

LEE, M. J. et al. Multi-agent based home network management system with extended real-time tuple space. In: INTERNATIONAL CONFERENCE ON INDUSTRIAL AND ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS, IEA/AIE 2004, 17., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2004. p.188 – 198. (Lecture Notes in Computer Science, v.3029).

LEIBNITZ, K.; WAKAMIYA, N.; MURATA, M. Biologically inspired self-adaptive multi-path routing in overlay networks. **Communications of the ACM**, New York, NY, USA, v.49, n.3, p.62–67, 2006.

LENG, X. et al. Connecting IPvX Networks Over IPvY with a P2P Method. **Journal of Network and Systems Management**, [S.l.], v.15, n.3, p.383–399, September 2007.

LI, Y. et al. Implementation of a cooperative network management framework. In: IEEE MILITARY COMMUNICATIONS CONFERENCE, 2001. MILCOM 2001. COMMUNICATIONS FOR NETWORK-CENTRIC OPERATIONS: CREATING THE INFORMATION FORCE. **Proceedings...** [S.l.: s.n.], 2001. p.611 – 615.

LI, Y.; SHOU, L.; TAN, K.-L. CYBER: a community-based search engine. In: EIGHTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING , 2008. P2P '08. **Proceedings...** [S.l.: s.n.], 2008. p.215–224.

LIMITED, S. **Skype**. Available at `http://www.skype.com/intl/en/welcomeback/`. Accessed in March 2010.

LIN, H.; CHENG, L. Modeling network bandwidth of IEEE 802.11 wireless local area networks. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 2005. **Proceedings...** [S.l.: s.n.], 2005. v.4, p.3563 – 3568.

LIN, P.; MACARTHUR, A.; LEANEY, J. Defining autonomic computing: a software engineering perspective. In: AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE, ASWEC'05., 2005. **Proceedings...** [S.l.: s.n.], 2005. p.88 – 97.

LIN, X. L. et al. Active Peer to Peer. In: SIXTH INTERNATIONAL CONFERENCE ON NETWORKING, 2007. ICN '07. **Proceedings...** [S.l.: s.n.], 2007. p.31 – 31.

LIU, B.; LI, W.; LUO, J. Agent cooperation in multi-agent based network management. In: THE 8TH INTERNATIONAL COMPUTER SUPPORTED COOPERATIVE WORK IN DESIGN, CSCWD 2004. **Proceedings...** [S.l.: s.n.], 2004. p.283 – 287.

LIU, F.; LI, Z. A Measurement and Modeling Study of P2P IPTV Applications. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE AND SECURITY, 2008. CIS '08. **Proceedings...** [S.l.: s.n.], 2008. v.1, p.114 – 119.

LIU, H. et al. An autonomic service architecture for self-managing grid applications. In: THE 6TH IEEE/ACM INTERNATIONAL WORKSHOP ON GRID COMPUTING, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.132 – 139.

LIU, H.; RILEY, G. How Efficient Peer-to-Peer Video Streaming Could Be? In: IEEE CONSUMER COMMUNICATIONS AND NETWORKING CONFERENCE, 2009. CCNC 2009., 6. **Proceedings...** [S.l.: s.n.], 2009. p.1 – 5.

LOHMAN, G. M.; LIGHTSTONE, S. S. SMART: making db2 (more) autonomic. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB '02, 28. **Proceedings...** VLDB Endowment, 2002. p.877 – 879.

LóPEZ, G. et al. A network access control approach based on the AAA architecture and authorization attributes. **Journal of Network and Computer Applications**, London, UK, UK, v.30, n.3, p.900–919, 2007.

LU, J.-L. et al. FISCO: a fully integrated scheme of self-configuration and self-organization for wsn. In: WCNC 2007. IEEE WIRELESS COMMUNICATIONS AND NETWORKING CONFERENCE, 2007., Kowloon. **Proceedings...** [S.l.: s.n.], 2007. p.3370 – 3375.

LUA J. CROWCROFT, R. C. E. keong et al. Securing Peer-to-Peer Content Sharing Service from Poisoning Attacks. In: EIGHTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING , 2008. P2P '08. **Proceedings...** [S.l.: s.n.], 2008. p.22–29.

LUO, J.; LI, W.; LIU, B. Distributed Intelligent Network Management Model for the Large-Scale Computer Network. In: INTERNATIONAL CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK IN DESIGN, CSCWD 2005, 9., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2006. p.313–323. (Lecture Notes in Computer Science, v.3865).

M., N. D. D. et al. Multiagent System Implementation for Network Management Based on SNMP Protocol. In: INTERNATIONAL SYMPOSIUM ON DISTRIBUTED COMPUTING AND ARTIFICIAL INTELLIGENCE 2008 (DCAI 2008), 2., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2008. p.38–46. (Advances in Soft Computing, v.50).

MALATRAS, A.; HADJIANTONIS, A. M.; PAVLOU, G. Exploiting Context-Awareness for the Autonomic Management of Mobile Ad Hoc Networks. **Journal of Network and Systems Management**, [S.l.], v.15, n.1, p.29–55, March 2007.

MALATRAS, A.; PAVLOU, G. A Practical Framework to Enable the Self-Management of Mobile Ad-Hoc Networks. In: GIIS 2007. FIRST INTERNATIONAL GLOBAL INFORMATION INFRASTRUCTURE SYMPOSIUM, 2007., Marrakech. **Proceedings...** [S.l.: s.n.], 2007. p.58 – 65.

MANZALINI, A. et al. **D1.1 Report on state-of-art, requirements and ACE model.** Available at `http://www.cascadas-project.org/deliv_m12.php`. Accessed in March 2010.

MANZALINI, A.; ZAMBONELLI, F. Towards Autonomic and Situation-Aware Communication Services: the cascadas vision. In: IEEE WORKSHOP ON DISTRIBUTED INTELLIGENT SYSTEMS: COLLECTIVE INTELLIGENCE AND ITS APPLICATIONS, 2006. DIS 2006. **Proceedings...** [S.l.: s.n.], 2006. p.383 – 388.

MARTIN-FLATIN, J.-P.; ZNATY, S.; HABAUX, J.-P. A Survey of Distributed Entreprise Network and Systems Management Paradigms. **Journal of Network and Systems Management**, [S.l.], v.7, n.1, p.9–26, 1999.

MATHIEU, B. et al. Self-Management of Context-Aware Overlay Ambient Networks. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2007. IM '07., 10. **Proceedings...** [S.l.: s.n.], 2007. p.749 – 752.

MATHIEU, B.; PARIS, P. A topology-aware P2P video streaming system. In: GLOBAL INFORMATION INFRASTRUCTURE SYMPOSIUM, 2009. GIIS '09. **Proceedings...** [S.l.: s.n.], 2009. p.1–8.

MCCLOGHRIE, K.; ROSE, M. **Management Information Base for Network Management of TCP/IP-based internets:mib-ii**. Available at `http://www.rfc-editor.org/rfc/std/std17.txt`. Accessed in March 2010., Internet Engineering Task Force (IETF), RFC 1213, STD 0017.

MCKINLEY, P. et al. Service Clouds: a distributed infrastructure for constructing autonomic communication services. In: IEEE INTERNATIONAL SYMPOSIUM ON DEPENDABLE, AUTONOMIC AND SECURE COMPUTING, 2. **Proceedings...** [S.l.: s.n.], 2006. p.341 – 348.

MEER, S. van der et al. Autonomic Networking: prototype implementation of the policy continuum. In: THE 1ST INTERNATIONAL WORKSHOP ON BROADBAND CONVERGENCE NETWORKS, 2006. BCN 2006. **Proceedings...** [S.l.: s.n.], 2006. p.1 – 10.

MEER, S. van der et al. Emerging Principles of Autonomic Network Management. In: MACE 2006: PROCEEDINGS OF THE FIRST IEEE INTERNATIONAL WORKSHOP ON MODELLING AUTONOMIC COMMUNICATIONS ENVIRONMENTS, Dublim, Irland. **Proceedings...** Multicon verlag, 2006. p.29–48.

MEULPOLDER, M. et al. Modeling and analysis of bandwidth-inhomogeneous swarms in BitTorrent. In: IEEE NINTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2009. P2P '09. **Proceedings...** [S.l.: s.n.], 2009. p.232 – 241.

MIHAILOVIC, A. et al. Building Knowledge Lifecycle and Situation Awareness in Self-Managed Cognitive Future Internet Networks. In: FIRST INTERNATIONAL CONFERENCE ON EMERGING NETWORK INTELLIGENCE, 2009. **Proceedings...** [S.l.: s.n.], 2009. p.3 – 8.

MILOJICIC, D. S. et al. **Peer-to-Peer Computing**. Available at `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.8222&rep=rep1&type=pdf`. Accessed in August 2010., Technical Report HPL-2002-57, HP Lab.

MININOVA. **Mininova**. Available at `http://www.mininova.org`. Accessed in March 2010.

MIORANDI, D. et al. **BIONETS - WP 1.1 REQUIREMENTS AND ARCHITECTURAL PRINCIPLES - D1.1.1 Application Scenario Analysis, Network Architecture Requirements and High-Level Specications**. Available at `http://www.bionets.eu/index.php?area=17`. Accessed in March 2010.

MIYAMURA, T. et al. Enhancing Bandwidth on Demand Service Based on Virtual Network Topology Control. In: IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM WORKSHOPS, 2008. NOMS WORKSHOPS 2008. **Proceedings...** [S.l.: s.n.], 2008. p.201–206.

MOURA, G. et al. On the Performance of Web Services Management Standards - An Evaluation of MUWS and WS-Management for Network Management. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2007. IM '07., 10. **Proceedings...** [S.l.: s.n.], 2007. p.459 – 468.

MULLER, N. J. Web-accessible network management tools. **International Journal of Network Management**, New York, NY, USA, v.7, n.5, p.288–297, 1997.

MURESANO, R.; REXACHS, D.; LUQUE, E. How SPMD applications could be efficiently executed on multicore environments? In: IEEE INTERNATIONAL CONFERENCE ON CLUSTER COMPUTING AND WORKSHOPS, 2009. CLUSTER '09. **Proceedings...** [S.l.: s.n.], 2009. p.1 – 4.

MURPHY, M. A. et al. Autonomic Clouds on the Grid. **Journal of Grid Computing**, [S.l.], v.8, n.1, p.1–18, March 2010.

NETWORKS, S. **KaZaA**. Available at `http://www.kazaa.com/`. Accessed in March 2010.

NGUENGANG, G. et al. **FP7 Information & Communication Technologies (ICT) - Self-NET - Deliverable D2.1**: first report on mechanisms for situation awareness of cognitive network elements and decision making mechanisms for goal-oriented task planning. Available at `https://www.ict-selfnet.eu/public-documents/deliverables`. Accessed in March 2010.

NIEBERT, N. et al. Network Virtualization: a viable path towards the future internet. **Journal Wireless Personal Communications**, [S.l.], v.45, n.4, p.511–520, June 2008.

NOBRE, J. C.; GRANVILLE, L. Z. Consistency of States of Management Data in P2P-Based Autonomic Network Management. In: IFIP/IEEE INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS: OPERATIONS AND MANAGEMENT, DSOM 2009, 20., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2009. p.99–110. (Lecture Notes in Computer Science, v.5841).

NTARMOS, N.; TRIANTAFILLOU, P. AESOP: altruism-endowed self-organizing peers. In: THIRD INTERNATIONAL WORKSHOP ON DATABASES, INFORMATION SYSTEMS AND PEER-TO-PEER COMPUTING (DBISP2P 2005), Heidelberg, Germany. **Proceedings...** Springer Berlin, 2005. p.151–165. (Lecture Notes in Computer Science, v.3367).

OBJECT MANAGEMENT GROUP, I. **History of CORBA**. Available at `http://www.omg.org/gettingstarted/history_of_corba.htm`. Accessed in March 2010.

OETIKER, T. MRTG - The Multi Router Traffic Grapher. In: LISA '98: Proceedings of the 12th USENIX Conference on System administration, Berkeley, CA, USA. **Proceedings...** USENIX Association, 1998. p.141–148.

OHSITA, Y. et al. Gradually Reconfiguring Virtual Network Topologies Based on Estimated Traffic Matrices. In: PROCEEDINGS OF 26TH IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS. IEEE INFOCOM 2007. **Proceedings...** [S.l.: s.n.], 2007. p.2511–2515.

OLFATI-SABER, R.; FAX, J.; MURRAY, R. Consensus and Cooperation in Networked Multi-Agent Systems. **Proceedings of the IEEE**, [S.l.], v.95, n.1, p.215 – 233, January 2007.

OMNET++. **Welcome to the OMNeT++ Community!** Available at `http://www.omnetpp.org/`. Accessed in March 2010.

PANISSON, A. **Aplicação de técnicas de distribuição de carga em sistemas de gerenciamento de redes baseados em p2p**. Master thesis (In Portuguese). Available at `http://www.lume.ufrgs.br/handle/10183/11450`. Accessed in May 2010.

PANISSON, A. et al. Designing the Architecture of P2P-Based network Management Systems. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, ISCC'06, 11. **Proceedings...** [S.l.: s.n.], 2006. p.69–75.

PAPADAKIS, H. et al. Imbuing unstructured P2P systems with non-intrusive topology awareness. In: IEEE NINTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2009. P2P '09. **Proceedings...** [S.l.: s.n.], 2009. p.51–60.

PAVLOU, G. On the Evolution of Management Approaches, Frameworks and Protocols: a historical perspective. **Journal of Network and Systems Management**, [S.l.], v.15, n.4, p.425–445, December 2007.

PEERCAST.ORG. **Peercast.org - P2P broadcasting for everyone**. Available at `http://www.peercast.org/`. Accessed in March 2010.

PENG, Y. et al. PD-Agent: a flexible agent bdi model for autonomic computing. In: INTERNATIONAL CONFERENCE ON MEASURING TECHNOLOGY AND MECHATRONICS AUTOMATION, 2009. ICMTMA '09. **Proceedings...** [S.l.: s.n.], 2009. v.3, p.721 – 724.

PEREIRA, R. C.; GRANVILLE, L. Z. On the performance of COPS-PR and NETCONF in an integrated management environment for DiffServ-enabled networks. In: INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 2008. ICT 2008. **Proceedings...** [S.l.: s.n.], 2008. p.1 – 6.

PRIETO, A. et al. Decentralized In-Network Management for the Future Internet. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS WORKSHOPS, 2009. ICC WORKSHOPS 2009. **Proceedings...** [S.l.: s.n.], 2009. p.1 – 5.

PRIETO, A. G.; STADLER, R. A-GAP: an adaptive protocol for continuous network monitoring with accuracy objectives. **IEEE Transactions on Network and Service Management**, [S.l.], v.4, n.1, p.2–12, 2007.

RANGANATHAN, A.; CAMPBELL, R. Autonomic pervasive computing based on planning. In: INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING, 2004. **Proceedings...** IEEE Computer Society, 2004. p.80 – 87.

RAO, Z.; GHENNIWA, H.; SHAMI, A. AGeMoS: an agent-based generic monitoring approach for self-management systems. In: INTERNATIONAL CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK IN DESIGN, 2007. CSCWD 2007., 11. **Proceedings...** [S.l.: s.n.], 2007. p.452 – 457.

RAZZAQUE, M. A.; DOBSON, S.; NIXON, P. Cross-Layer Architectures for Autonomic Communications. **Journal of Network and Systems Management**, [S.l.], v.15, n.1, p.13–27, 2007.

RHEA, S. et al. Maintenance-free global data storage. **IEEE Internet Computing**, [S.l.], v.5, n.5, p.40 – 49, September/October 2001.

ROCHA, A. da; ROCHA, C. A. da; SOUZA, J. N. de. Script MIB Extension for Resource Limitation in SNMP Distributed Management Environments. In: INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS AND NETWORKING. ICT 2004, Heidelberg, Germany. **Proceedings...** Springer Berlin, 2004. p.835–840. (Lecture Notes in Computer Science, v.3124).

ROHR, E.; GRANVILLE, L.; TAROUCO, L. Evaluating WS-security and XACML in web services-based network management. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2009. IM '09. **Proceedings...** [S.l.: s.n.], 2009. p.188 – 194.

RUBIO-LOYOLA, J. et al. A viewpoint of the network management paradigm for Future Internet networks. In: IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT-WORKSHOPS, 2009. IM '09. **Proceedings...** [S.l.: s.n.], 2009. p.93 – 100.

SACHA, J. et al. Discovery of Stable Peers in a Self-organising Peer-to-Peer Gradient Topology. In: IFIP WG 6.1 INTERNATIONAL CONFERENCE ON DISTRIBUTED APPLICATIONS AND INTEROPERABLE SYSTEMS, DAIS 2006, 6., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2006. p.70–83. (Lecture Notes in Computer Science, v.4025).

SAMAAN, N.; KARMOUCH, A. Towards Autonomic Network Management: an analysis of current and future research directions. **IEEE Communications Surveys & Tutorials**, [S.l.], v.11, n.3, p.22–36, August 2009.

SAMIMI, F. et al. Service Clouds: distributed infrastructure for adaptive communication services. **IEEE Transactions on Network and Service Management**, [S.l.], v.4, n.2, p.84–95, 2007.

SANCHEZ-ARTIGAS, M.; GARCIA-LOPEZ, P. On routing in Distributed Hash Tables: is reputation a shelter from malicious behavior and churn? In: IEEE NINTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2009. P2P '09. **Proceedings...** [S.l.: s.n.], 2009. p.31–40.

SANGHAN, S.; HASAN, M. Intelligent P2P VoIP through Extension of Existing Protocols. In: THE 9TH INTERNATIONAL CONFERENCE ON ADVANCED COMMUNICATION TECHNOLOGY. **Proceedings...** [S.l.: s.n.], 2007. v.3, p.1597 – 1601.

SCHMIEG, A. et al. pSense - Maintaining a Dynamic Localized Peer-to-Peer Structure for Position Based Multicast in Games. In: EIGHTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING , 2008. P2P '08. **Proceedings...** [S.l.: s.n.], 2008. p.247–256.

SCHÖNWÄLDER, J. et al. **Minutes of the Joint EMANICS / IRTF-NMRG Workshop on Future Direction of Network and Service Management Research**. Available at `http://www.ibr.cs.tu-bs.de/projects/nmrg/minutes/minutes-021.txt`. Accessed in April 2010.

SCHÖNWÄLDER, J.; QUITTEK, J.; KAPPLER, C. Building distributed management applications with the IETF ScriptMIB. **IEEE Journal on Selected Areas in Communications**, [S.l.], v.18, n.5, p.702–714, 2000.

SCHUETZ, S. et al. Autonomic and Decentralized Management of Wireless Access Networks. **IEEE Transactions on Network and Service Management**, [S.l.], v.4, n.2, p.96–106, September 2007.

SCHULZE, B. et al. MomentA: service management using mobile agents in a corba environment. **Journal of Network and Systems Management**, [S.l.], v.9, n.2, p.203–222, 2001 June.

SEKERCIOGLU, Y. A.; IVANOVICH, M.; YEGIN, A. A survey of MAC based QoS implementations for WiMAX networks. **Computer Networks**, [S.l.], v.53, n.14, p.2517–2536, September 2009.

SERRANO, J.; SERRAT, J.; STRASSNER, J. Ontology-Based Reasoning for Supporting Context-Aware Services on Autonomic Networks. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 2007. ICC '07. **Proceedings...** [S.l.: s.n.], 2007. p.2097 – 2102.

SHEN, P. **TVU Networks - TV Without Limits**. Available at `http://tvunetworks.com/doc/whatis.html`. Accessed in March 2010.

SIMMONS, B.; LUTFIYYA, H. Policies, grids and autonomic computing. In: DEAS '05: PROCEEDINGS OF THE 2005 WORKSHOP ON DESIGN AND EVOLUTION OF AUTONOMIC APPLICATION SOFTWARE, New York, NY, USA. **Proceedings...** ACM Press, 2005. p.1–5.

SIMON, C. et al. Peer-to-Peer management in Anbient Networks. In: IST MOBILE AND WIRELESS COMMUNICATIONS SUMMIT 2005. **Proceedings...** [S.l.: s.n.], 2005.

SLOMN, M. Policy Driven Management For Distributed Systems. **Journal of Network and Systems Management**, [S.l.], v.2, n.4, p.333–360, December 1994.

SOLDATOS, J.; ALEXOPOULOS, D. Web services-based network management: approaches and the wsnet system. **International Journal of Network Management**, New York, NY, USA, v.17, n.1, p.33–50, 2007.

STADLER, R. **New Approaches to Distributed Management to Achieve Scalability and Robustness**. 19th Meeting of Network Management Research Group - IRTF. Available at `http://www.ibr.cs.tu-bs.de/projects/nmrg/meetings/2006/stockholm/nmrg-19-stadler.pdf`. Accessed in April 2010.

STANFEL, Z.; HOCENSKI, Z.; MARTINOVIC, G. A Self Manageable Rule Driven Enterprise Application. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY INTERFACES, 2007. ITI 2007., 29. **Proceedings...** [S.l.: s.n.], 2007. p.717 – 722.

STATE, R.; FESTOR, O. A Management Platform Over Peer-to-Peer Service Infrastructure. In: INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 2003. ICT 2003, 10., Vancouver, BC, Canada. **Proceedings...** [S.l.: s.n.], 2003. p.124–131.

STEINDER, M. et al. Server virtualization in autonomic management of heterogeneous workloads. In: PROCEEDINGS OF 10TH IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT (IM 2007). **Proceedings...** [S.l.: s.n.], 2007. p.139–148.

STEPHAN, R.; RAY, P.; PARAMESH, N. Network management platform based on mobile agents. **International Journal of Network Management**, New York, NY, USA, v.14, n.1, p.59–73, 2004.

STERRITT, R.; BUSTARD, D. Autonomic Computing - a means of achieving dependability? In: IEEE INTERNATIONAL CONFERENCE AND WORKSHOP ON THE ENGINEERING OF COMPUTER-BASED SYSTEMS, 2003., 10. **Proceedings...** [S.l.: s.n.], 2003. p.247 – 251.

STERRITTA, R. et al. A concise introduction to autonomic computing. **Advanced Engineering Informatics**, [S.l.], v.19, n.3, p.181–187, July 2005.

STOICA, I. et al. Chord: a scalable peer-to-peer lookup protocol for internet applications. **IEEE/ACM Transactions on Networking**, [S.l.], v.11, n.1, p.17 – 32, February 2003.

STRASSNER, J. DEN-ng: achieving business-driven network management. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 2002. NOMS 2002., 2002. **Proceedings...** [S.l.: s.n.], 2002. p.753 – 766.

STRASSNER, J. C.; AGOULMINE, N.; LEHTIHET, E. FOCALE - A Novel Autonomic Network Architecture. In: LAACS 2006: PROCEEDINGS OF THE 1ST LATIN AMERICAN AUTONOMIC COMPUTING SYMPOSIUM, Campo Grande, BR. **Proceedings...** [S.l.: s.n.], 2006. p.48–60.

STRASSNER, J. et al. Modelling Context for Autonomic Networking. In: IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM WORKSHOPS, 2008. NOMS WORKSHOPS 2008. **Proceedings...** [S.l.: s.n.], 2008. p.299 – 308.

STRASSNER, J. et al. The Design of a New Policy Model to Support Ontology-Driven Reasoning for Autonomic Networking. **Journal of Network and Systems Management**, [S.l.], v.17, n.1-2, p.5–32, June 2009.

STRAUSS, F.; SCHÖNWÄLDER, J.; QUITTEK, J. Open source components for Internet management by delegation. In: IEEE/IFIP INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT PROCEEDINGS, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.185 – 198.

TANENBAUM, A. S.; STEEN, M. V. (Ed.). **Distributed Systems - Principles and Designs**. 2.ed. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2007.

TANG, Y.; AL-SHAER, E. Towards Collaborative User-Level Overlay Fault Diagnosis. In: IEEE INFOCOM 2008. THE 27TH CONFERENCE ON COMPUTER COMMUNICATIONS. **Proceedings...** [S.l.: s.n.], 2008. p.2476 – 2484.

TEAM eMule. **eMule**. Available at `http://www.emule-project.net/home/perl/general.cgi?l=1`. Accessed in March 2010.

TEAM', O. **Osiris Serverless Portal System**. Available at `http://osiris.kodeware.net/index.php`. Accessed in March 2010.

TERAUCHI, A.; AKASHI, O. Trust-Based Cooperative Action Control in Multi-agent Systems for Network Management. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS, 2009. WAINA '09. **Proceedings...** [S.l.: s.n.], 2009. p.278 – 283.

TESAURO, G. et al. A Multi-Agent Systems Approach to Autonomic Computing. In: THIRD INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS '04, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2004. p.464–471.

TIMM, I. J. et al. From Agents to Multiagent Systems. In: KIRN, S. et al. (Ed.). **Multiagent Engineering - Theory and Applications in Enterprises**. Heidelberg, Germany: Springer Berlin, 2006. p.35–51. (International Handbooks on Information Systems).

TIRASOONTORN, K.; KAMOLPHIWONG, S.; SAE-WONG, S. Distributed P2P-SIP conference construction. In: MOBILITY '08: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON MOBILE TECHNOLOGY, APPLICATIONS, AND SYSTEMS, New York, NY, USA. **Proceedings...** ACM, 2008. p.1–5.

TIZGHADAM, A.; LEON-GARCIA, A. AORTA: autonomic network control and management system. In: IEEE CONFERENCE ON COMPUTER COMMUNICATIONS WORKSHOPS, 2008. INFOCOM., Rio de Janeiro, Brazil. **Proceedings...** [S.l.: s.n.], 2008. p.1 – 4.

TIZGHADAM, A.; LEON-GARCIA, A. Autonomic traffic engineering for network robustness. **IEEE Journal on Selected Areas in Communications**, New York, USA, v.28, n.1, p.39–50, January 2010.

TO, H. H.; KRISHNASWAMY, S.; SRINIVASAN, B. Mobile agents for network management: when and when not! In: SAC '05: PROCEEDINGS OF THE 2005 ACM SYMPOSIUM ON APPLIED COMPUTING, New York, NY, USA. **Proceedings...** ACM, 2005. p.47–53.

TRIMINTZIOS, P. et al. DiMAPI: an application programming interface for distributed network monitoring. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 2006. NOMS 2006., 10. **Proceedings...** [S.l.: s.n.], 2006. p.382 – 393.

TRIPATHI, S.; HUANG, Y.; JAJODIA, S. Local Area Networks: software and related issues. **IEEE Transactions on Software Engineering**, [S.l.], v.SE-13, n.8, p.872 – 879, August 1987.

TRUMLER, W. et al. AMUN - autonomic middleware for ubiquitous environments applied to the smart doorplate project. In: INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING, ICAC 2004. **Proceedings...** IEEE Computer Society, 2004. p.274 – 275.

TRUMLER, W. et al. AMUN: an autonomic middleware for the smart doorplate project. **Personal Ubiquitous Computing**, London, UK, v.10, n.1, p.7–11, 2005.

TRZEC, K.; HULJENIC, D. Intelligent agents for QoS management. In: AAMAS '02: PROCEEDINGS OF THE FIRST INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, New York, NY, USA. **Proceedings...** ACM, 2002. p.1405–1412.

UNIVERSITY, S.; RIVERSIDE, U. **Quake-Catcher Network**. Available at `http://boinc.bakerlab.org/rosetta/`. Accessed in March 2010.

VARVELLO, M.; DIOUT, C.; BIERSACK, E. P2P Second Life: experimental validation using kad. In: IEEE INFOCOM 2009. THE 28TH CONFERENCE ON COMPUTER COMMUNICATIONS. **Proceedings...** [S.l.: s.n.], 2009. p.1161 – 1169.

VIANNA, R. et al. Evaluating the Performance of Web Services Composition for Network Management. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 2007. ICC '07. **Proceedings...** [S.l.: s.n.], 2007. p.1943 – 1948.

VIDALES, P. et al. Autonomic system for mobility support in 4G networks. **IEEE Journal on Selected Areas in Communications**, [S.l.], v.23, n.12, p.2288 – 2304, December 2005.

VIROLI, M.; CASADEI, M.; OMICINI, A. A framework for modelling and implementing self-organising coordination. In: SAC '09: PROCEEDINGS OF THE 2009 ACM SYMPOSIUM ON APPLIED COMPUTING, New York, NY, USA. **Proceedings...** ACM, 2009. p.1353–1360.

WACKER, A. et al. A NAT Traversal Mechanism for Peer-To-Peer Networks. In: EIGHTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING , 2008. P2P '08. **Proceedings...** [S.l.: s.n.], 2008. p.81–83.

WANG, C. et al. Self-Organizing Content Distribution in a Data Indexed DHT Network. In: SIXTH IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2006. P2P 2006. **Proceedings...** [S.l.: s.n.], 2006. p.241 – 248.

WANG, X. Y. et al. A Resource Management Framework for Multi-tier Service Delivery in Autonomic Virtualized Environments. In: NOMS 2008. IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 2008., Salvador, Bahia. **Proceedings...** [S.l.: s.n.], 2008. CDROM.

WANG, Y. et al. Virtual routers on the move: live router migration as a network-management primitive. In: SIGCOMM '08: PROCEEDINGS OF THE ACM SIGCOMM 2008 CONFERENCE ON DATA COMMUNICATION, New York, NY, USA. **Proceedings...** ACM, 2008. p.231–242.

WANG, Y.; LI, X.-Y.; ZHANG, Q. Efficient Self Protection Algorithms for Static Wireless Sensor Networks. In: GLOBECOM '07. IEEE GLOBAL TELECOMMUNICATIONS CONFERENCE, 2007., Washington, DC. **Proceedings...** [S.l.: s.n.], 2007. p.931 – 935.

WEISS, G. (Ed.). **Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence**. [S.l.]: MIT Press, 1999.

WESTERINEN, A. et al. **RFC 3198 - Terminology for Policy-Based Management**. Available at `http://www.rfc-editor.org/rfc/rfc3198.txt`. Accessed in March 2010.

WOLF, T.; HOLVOET, T. Towards autonomic computing: agent-based modelling, dynamical systems analysis, and decentralised control. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS, 2003. INDIN 2003. **Proceedings...** [S.l.: s.n.], 2003. p.470 – 479.

WRIGHT, M. J. Using policies for effective network management. **International Journal of Network Management**, New York, NY, USA, v.9, n.2, p.118–125, 1999.

XHAFA, F. et al. Efficient Peer Selection in P2P JXTA-Based Platforms. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS, 2008. AINA 2008., 22. **Proceedings...** [S.l.: s.n.], 2008. p.1013 – 1020.

XIE, H.; MIN, B.; DAI, Y. SODA: towards a framework for self optimization via demand adaptation in peer-to-peer networks. In: IEEE NINTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, 2009. P2P '09. **Proceedings...** [S.l.: s.n.], 2009. p.163–170.

XU, H. et al. On Network Management Solutions and the Potential of P2P Technologies. In: SECOND INTERNATIONAL CONFERENCE ON FUTURE GENERATION COMMUNICATION AND NETWORKING, 2008. FGCN '08. **Proceedings...** [S.l.: s.n.], 2008. v.1, p.403 – 406.

XU, H.; SONG, M.; LUO, X. A QoS-oriented management framework for reconfigurable mobile mashup services. In: INTERNATIONAL CONFERENCE ON ADVANCED COMMUNICATION TECHNOLOGY, 2009. ICACT 2009., 11. **Proceedings...** [S.l.: s.n.], 2009. v.3, p.2001 – 2005.

YALAGANDULA, P. et al. S3: a scalable sensing service for monitoring large networked systems. In: INM '06: PROCEEDINGS OF THE 2006 SIGCOMM WORKSHOP ON INTERNET NETWORK MANAGEMENT, New York, NY, USA. **Proceedings...** ACM Press, 2006. p.71–76.

YU, M. et al. Rethinking virtual network embedding: substrate support for path splitting and migration. **SIGCOMM Computer Communications Review**, New York, NY, USA, v.38, n.2, p.17–29, 2008.

ZEWDU, Z.; DENKO, M.; LIBSIE, M. Workload Characterization of Autonomic DBMSs Using Statistical and Data Mining Techniques. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS, 2009. WAINA '09. **Proceedings...** [S.l.: s.n.], 2009. p.244 – 249.

ZHANG, J. et al. A New Conceptual Distributed Network Measurement Architecture Based on Web Service. In: IEEE INTERNATIONAL CONFERENCE ON NETWORKS, 2006. ICON '06., 14. **Proceedings...** [S.l.: s.n.], 2006. v.2, p.1 – 7.

ZHANG, R. Autonomic Performance Recuperation for Service-oriented Systems. In: IEEE INTERNATIONAL CONFERENCE ON SERVICES COMPUTING, 2007. SCC 2007. **Proceedings...** [S.l.: s.n.], 2007. p.544 – 551.

ZHANG, W.; HANSEN, K. M. Semantic Web Based Self-Management for a Pervasive Service Middleware. In: SASO '08. SECOND IEEE INTERNATIONAL CONFERENCE ON SELF-ADAPTIVE AND SELF-ORGANIZING SYSTEMS, 2008., Venezia. **Proceedings...** [S.l.: s.n.], 2008. p.245 – 254.

ZHANG, X. et al. Using P2P Overlay to Improve VoIP Quality in SIP+P2P System. In: WASE INTERNATIONAL CONFERENCE ON INFORMATION ENGINEERING, 2009. ICIE '09. **Proceedings...** [S.l.: s.n.], 2009. p.255 – 259.

ZHANG, Y.-Y. et al. An Energy-Efficient Multi-agent Based Architecture in Wireless Sensor Network. In: ASIA-PACIFIC WEB CONFERENCE ON PROGRESS IN WWW RESEARCH AND DEVELOPMENT, APWEB 2008, 10., Heidelberg, Germany. **Proceedings...** Springer Berlin, 2008. p.124–129. (Lecture Notes in Computer Science, v.4976).

ZHANG, Z.; LI, B. Neighbor discovery in mobile ad hoc self-configuring networks with directional antennas: algorithms and comparisons. **Wireless Communications, IEEE Transactions on**, [S.l.], v.7, n.5, p.1540–1549, May 2008.

ZHAO, Z.; CHEN, J.; CRESPI, N. A policy-based framework for autonomic reconfiguration management in heterogeneous networks. In: MUM '08: PROCEEDINGS OF THE 7TH INTERNATIONAL CONFERENCE ON MOBILE AND UBIQUITOUS MULTIMEDIA, New York, NY, USA. **Proceedings...** ACM, 2008. p.71–78.

ZHENG, G. ping; DONG, W. bo. The research of mobile agent-based distributed network management. In: ISECS INTERNATIONAL COLLOQUIUM ON COMPUTING, COMMUNICATION, CONTROL, AND MANAGEMENT, 2009. CCCM 2009. **Proceedings...** [S.l.: s.n.], 2009. p.182 – 185.

ZHOU, L.; RENESSE, R. van. P6P: a peer-to-peer approach to internet infrastructure. **Lecture Notes in Computer Science - Peer-to-Peer Systems III**, Heidelberg, Germany, v.3279, p.75–86, January 2005.

ZHOU, Y.; LYU, M. R. An Energy-Efficient Mechanism for Self-Monitoring Sensor Web. In: IEEE AEROSPACE CONFERENCE, 2007., Big Sky, MT, USA. **Proceedings...** [S.l.: s.n.], 2007. p.1 – 8.

ZINNER, T. et al. On the Trade-Off between Efficiency and Congestion in Location-Aware Overlay Networks - Example of a Vertical Handover Support System. In: EIGHTH INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING , 2008. P2P '08. **Proceedings...** [S.l.: s.n.], 2008. p.237–246.

ZOMAYA, A. Y. H. (Ed.). **Parallel & Distributed Computing Handbook**. New York, NY, USA: McGraw-Hill, Inc., 1996.

ZSEBY, T. et al. Towards a Future Internet: node collaboration for autonomic communication. **Towards the Future Internet - A European Research Perspective**, Amsterdan, Netherlands, v.0, p.123 – 135, 2009.

APPENDIX A - DETAILS OF SELF-HEALING P2P

This appendix describes some relevant details of the developed solution on the self-healing P2P approach. It will be presented the diagram class of the implementation of such approach based on the ManP2P platform. Next, the details of some important group communication mechanisms for the self-healing P2P solution are illustrate and described.

A.1 Class Diagram of Self-healing Support on ManP2P Platform

The class diagram presented in Figure A.1 illustrates a partial view of the implementation of the ManP2P platform. The portion depicted in this figure presents the implementation associated with the self-healing P2P support for the management services of the ManP2P platform. The details about the class diagrams related solely to the ManP2P are described by Panisson (PANISSON, 2007).

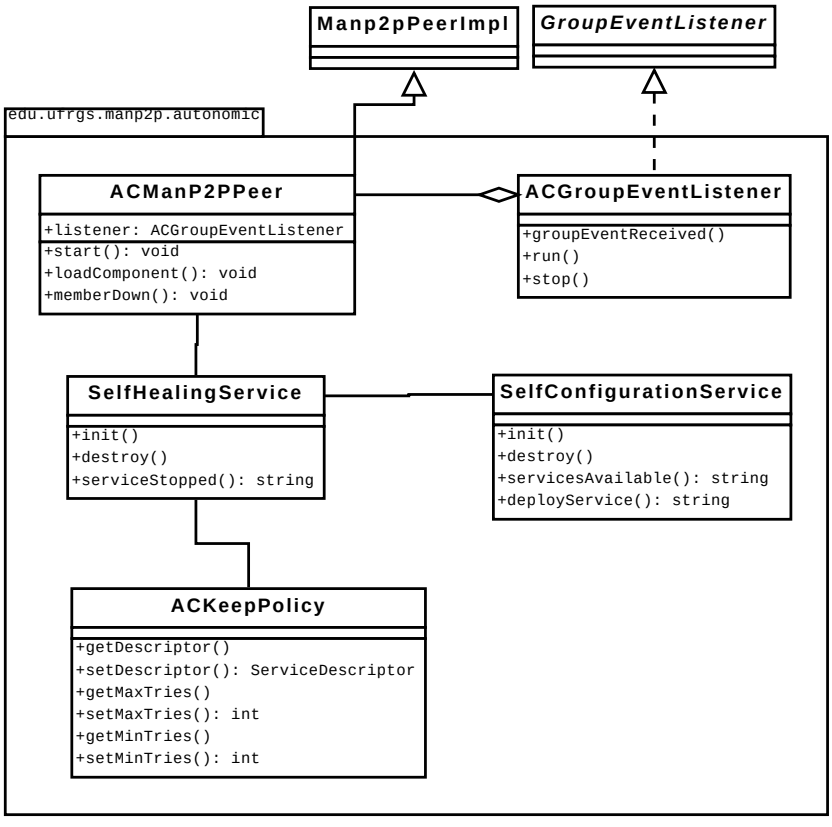


Figure A.1: Class diagram of self-healing support inside ManP2P platform

The *edu.ufrgs.manp2p.autonomic* package encapsulates the classes that support the self-healing P2P behavior inside the ManP2P platform. Each peer inside the platform is able to start up the self-healing support whenever it is configured to instantiate the *ACManP2PPeer* class. The objective of this class is to be the middle element between the core of the ManP2P platform and the services associated with the self-healing P2P support. The *listener* attribute (that is an object of *ACGroupEventListener*) is responsible to receive messages that arrive from each management peer group. To achieve this, the *loadComponent()* method of the *ACManP2PPeer* class register the *listener* attribute to hear the messages associated to each management peer group configured inside the peer. The process of listening for messages is handled by the *memberDown()* method of *ACManP2PPeer* class. Thus, whenever a member down message is recognized, the *SelfHealingService* is activated.

The *ACKeepPolicy* class is responsible to handle the information of the policies that describe how many instances of each management service should be running inside of the management overlay. This way, before starting a recovery process a *SelfHealingService* object gets information from the *ACKeepPolicy* object. After this, if necessary, the *SelfHealingService* object contacts the *SelfConfigurationService* object asking it to search for an available peer and deploy a new instance of the failed management service. In the next section, the low level interactions among peers and the objects inside the peers are described.

A.2 Description of Internal Aspects of Self-healing Algorithms

Some operations of the self-healing P2P proposal lie on the group communication mechanism of the management P2P overlay. The important operations for the execution of the solution proposed are described in this section. In order to better describe those operations, Figure A.2 depicts a scenario of the self-healing P2P employment. Furthermore, this figure illustrates one manner to visualize the existence of the management peer groups. This view is important to understand the group operations.

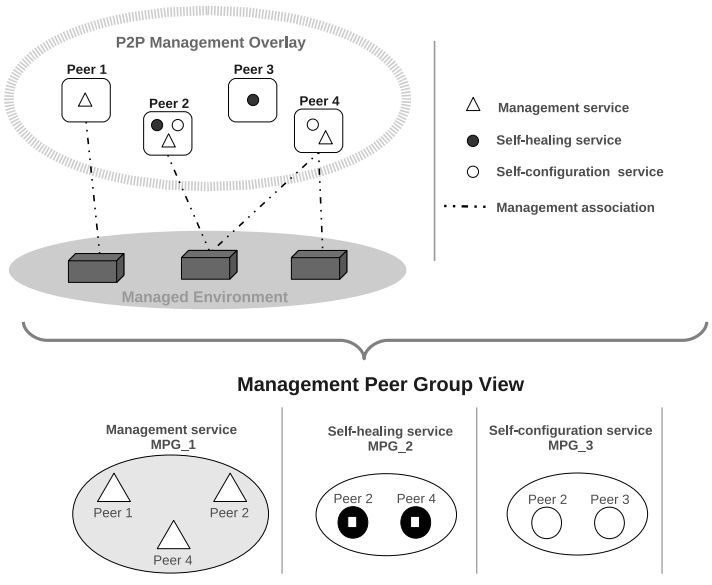


Figure A.2: Example of management peer group view

The environment presented in Figure A.2 shows four peers and three services: management service (the triangle figure), self-healing service (the black circle), and the self-configuration service (the white circle). The management peer group view illustrated in the bottom of the figure gives the idea how those services are organized inside the management overlay. This way, MPG_1 is related to the management service, MPG_2 to the self-healing service, and MPG_3 to the self-configuration service. The next figures use the management peer group illustration to detail the operations of identifying and recovering a failed service.

Starting with the identification of failures, Figure A.3 shows the important internal details of the peers involved in the identification of a failure. As mentioned in Chapter 4, the failure identification happens embedded in the group communication mechanism. For this reason Figure A.3 shows a diagram with the peers associated with the MPG_1, where a failure on an instance of a service happened and has to be detected, as illustrated.

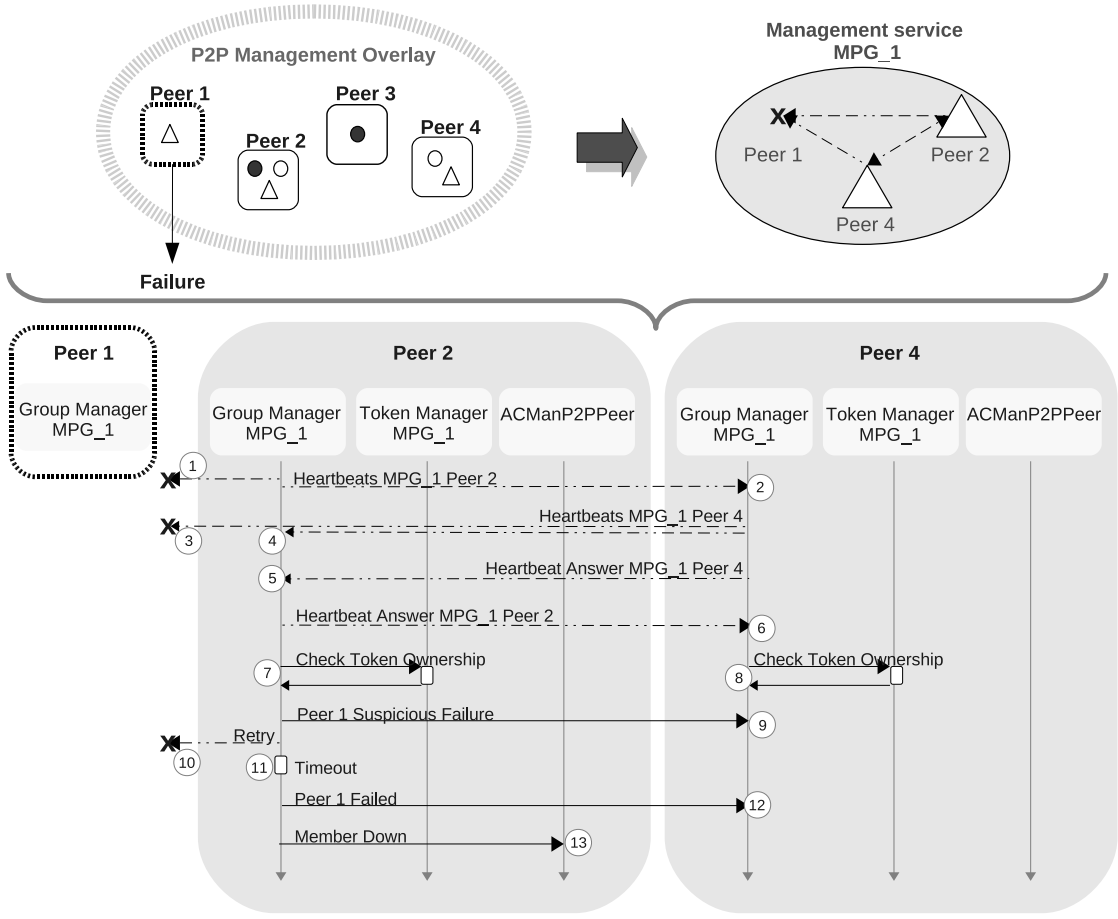


Figure A.3: Failure identification diagram

The steps 1-6 of Figure A.3 show the exchange of heartbeats inside the peers of the management peer group MPG_1. In this example, it is not considered whether the group communication mechanism uses multicast on the network level or at the application level. The relevant fact is that messages are exchanged among the members of the peer group. In Figure A.3, the messages on the steps 1 and 3 are lost, and the answer for the heartbeats of Peer 1 never arrive for the other peers of the management group MPG_1. At this point, the Group Manager module of each peer (in this example, Peer 2 and 4) check with the

Token Manager module of each peer if they are the owner of the token (steps 7 and 8 of Figure A.3). Only the owner of the token is able to proceed with the failure identification operation. In this example, Peer 2 is the owner of the token associated with MPG_1. Thus, before declaring the instance of the management service of MPG_1 inside Peer 1 as failed, the Group Manager of Peer 2 declares it as suspicious of failure. All the other peers of the MPG_1 are informed by the owner of the token (step 9 of Figure A.3) about the suspicious situation, and the owner retries the contact with the suspicious peer (step 10 of Figure A.3). After a configurable timeout (step 11 of Figure A.3), if no answer is heard from the suspicious peer, then the owner of the token gets in contact with the other members of the MPG_1 announcing the failure (step 12 of Figure A.3). Finally, the owner of the token also invokes the *memberDown()* operation of the ACManP2PPeer module announcing that a member of MPG_1 is down and some verification must be performed (step 12 of Figure A.3). The internal details of the recovery process are depicted in Figure A.4.

As presented in the previous section, the class diagram of the self-healing P2P solution shows that there is an association between the ACManP2PPeer module and the self-healing service. In fact, whenever such module receives the member down message, it invokes the evaluation operation of the self-healing service. Due to the fact that the self-healing service also forms a management peer group, *i.e.* MPG_2 in the example of Figure A.4, all the peers inside MPG_2 will receive the message that an instance of some management group is down (step 1 of Figure A.4). Once again, only the owner of the token associated with MPG_2 will proceed with the evaluation process (steps 2-4 of Figure A.4).

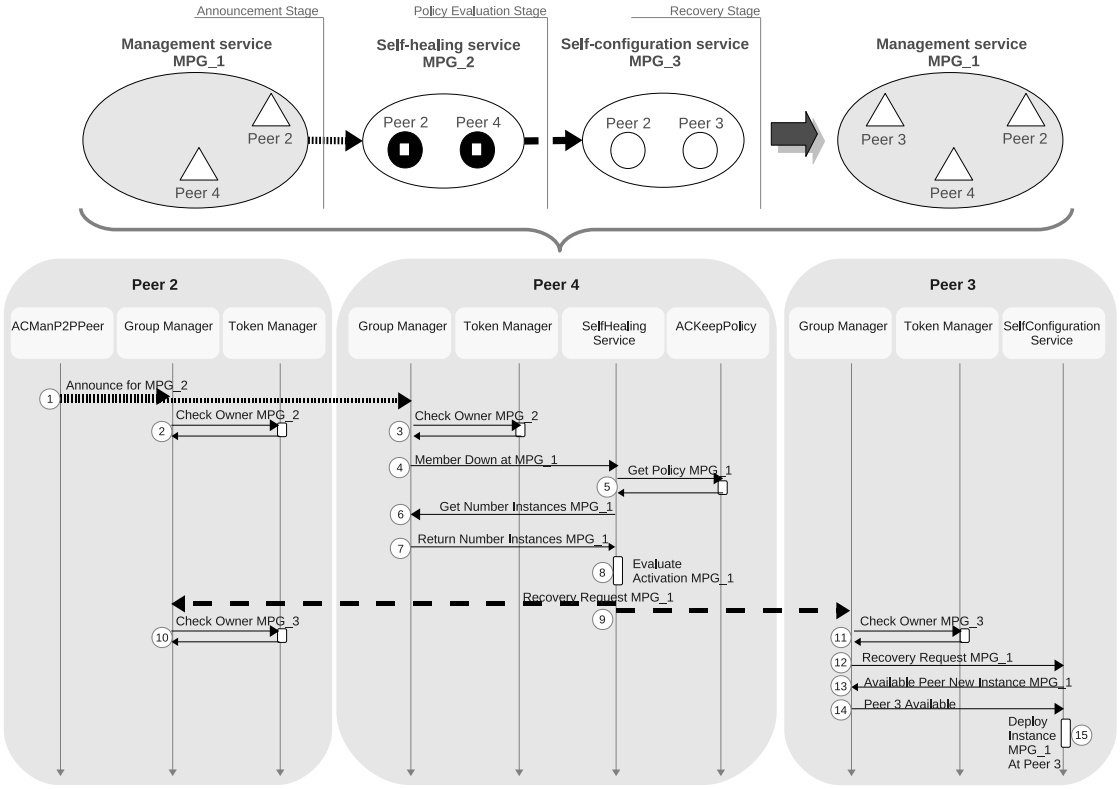


Figure A.4: Failure recovery diagram

The first actions of the self-healing service instance owning the token are: (i) verify

the terms of the policy associated with the management service in failure, step 5 of Figure A.4; (ii) verify how many instances of the failed management service are currently deployed inside the management overlay, steps 6 and 7 of Figure A.4. Based on the retrieved information, the self-healing service is able to evaluate the need of a recovery, step 8 of Figure A.4. In case of a positive evaluation for a new deployment, the self-healing service will invoke the deployment operation of the self-configuration service. In the example depicted in this section, Peer 4 that owns the token of the MPG_2 sends a recovery request for the members of MPG_3, that in this case are hosted by Peers 2 and 3 (step 9 of Figure A.4). The ownership of the token associated with the MPG_3 is evaluated (steps 10 and 11 of Figure A.4) and Peer 3, *i.e.*, the owner of the token, contacts peers in order to discover an available peer to deploy an instance of the management service failed in the MPG_1 (steps 12-14 of Figure A.4). If an available peer is found, then an instance of the same type of failed management service is deployed. In this example, Peer 3 is the available peer receiving the new instance of the management service of MPG_1, healing the failure of such management peer group in step 15 of Figure A.4.

APPENDIX B - DETAILS OF SELF-ORGANIZING P2P

This appendix aims at describing some relevant details of the developed solution on the self-organizing P2P approach. It is presented the virtualization module developed on Omnet++ to simulate the proposed approach and also the details of the monitoring process that is important for the decision-making mechanism.

B.1 Network Virtualization Module for Omnet++

The definition of a new module for Omnet++ starts with the identification of the elements and infrastructures that compose the simulation environment. In Omnet++, a module is composed of the specification of the simulation elements inside NED files and the implementation of such elements using the Omnet++ library support and the C++ language. It is out of the scope of this section describe the details of Omnet++ simulator, and further information can be found under the online documentation (OMNET++, 2010). The network virtualization model developed is based on the Omnet++ 3.4b2 version.

There are two basic types of modules that can be defined in Omnet++ 3.4b2: compound (identified by the *"Module"* term in the NED file) and simple modules (identified by the term *"simple"*). The former can be composed of submodules, parameters, and gates, while the latter can be composed of parameters and gates. The users' of Omnet++ are free to implement the simple modules defined inside the NED files, but the compound modules are automatically generated by the Omnet++ simulator. Figure B.1 illustrates the compound and simple modules defined for the network virtualization module.

An hierarchy of modules is established in Figure B.1. The first level shows the *SelfOrganizingVirtualNetwork* module that corresponds to the simulation environment. This means that the network environment to be simulated can be composed of *PhysicalNode* modules, *RequestFactory* and *RequestFactoryManager*. Those three modules are in the second level of the hierarchy. It is possible to instantiate more than one *PhysicalNode* module inside the *SelfOrganizingVirtualNetwork* scope. However, there must be only one instance of *RequestFactory* and *RequestFactoryManager* inside the network environment. Figure B.2 shows the graphical interface of Omnet++ and presents how the second level of the hierarchy of modules is instantiated. The *RequestFactory* and *RequestFactoryManager* modules are used to determine the request rate of the transmission among the physical nodes, while the *PhysicalNode* module encloses the core elements of the network virtualization environment.

Inside the *PhysicalNode* there are the modules that compose the third level of the hierarchy: *VirtualManager*, *Storage*, *SelfOrganization*, *VirtualEntitiesMonitor*, *VirtualNode*, and *VirtualPipe*. The elements illustrate in Figure B.1 with the "*" mark are not instantiated directly in the NED file, but they are deployed dynamically according to parameters

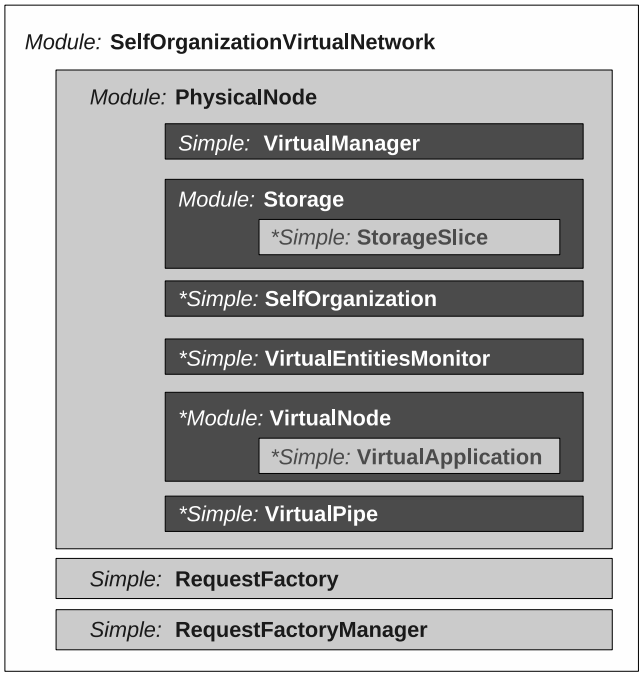


Figure B.1: Illustration of the elements composing the network virtualization module

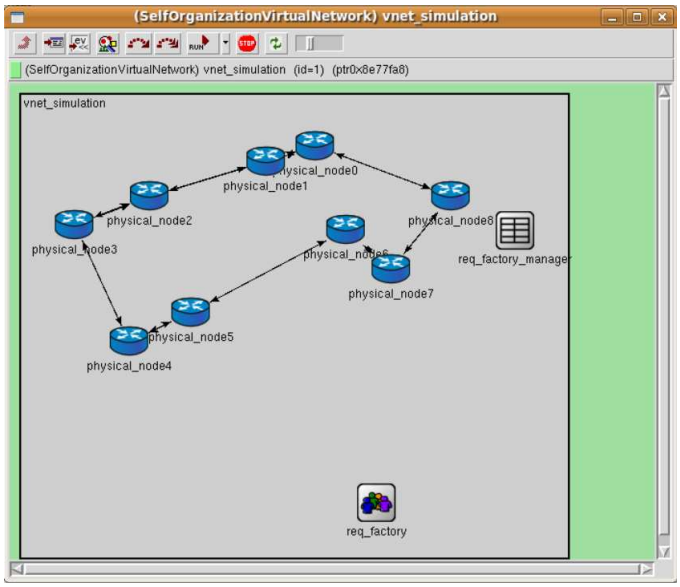
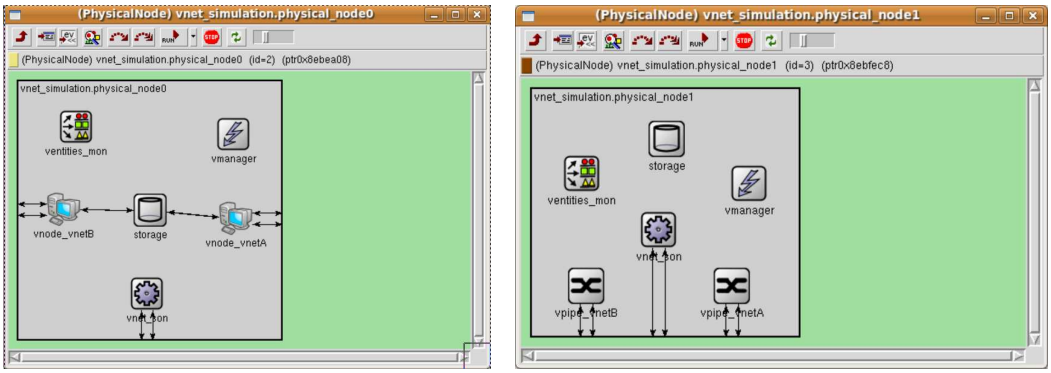


Figure B.2: Instantiation of the network virtualization module

during the startup phase of the simulation. The *VirtualNode* and *VirtualPipe* modules are optional and the others are mandatory inside the *PhysicalNode* module. Figure B.3 depicts how the third level modules are organized.

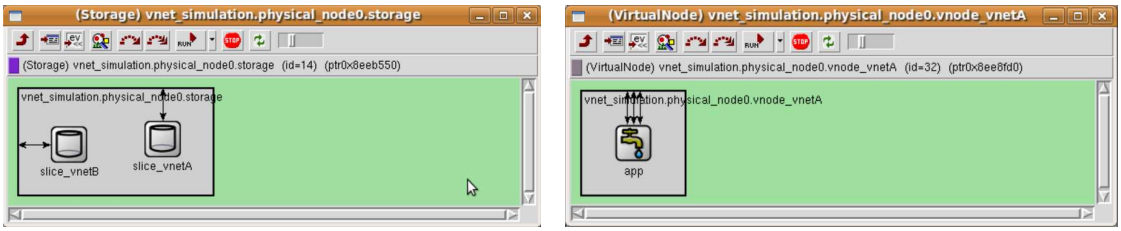
The *VirtualManager* and *SelfOrganization* modules are the core elements for the self-organizing P2P approach proposed in this thesis. Based on the information extracted from the *VirtualEntitiesMonitor* the core elements can perform their tasks. The details of the monitoring process are presented later on the text.



(a) Physical Node 0 - Virtual Node (b) Physical Node 1 - Virtual Pipe

Figure B.3: Virtual elements representation inside physical nodes

Finally, the forth level of the hierarchy of modules regards the *VirtualNode* and *Storage*modules. The simple modules *VirtualApplication* and *StorageSlice* exist whenever there is a *VirtualNode* module deployed inside the *PhysicalNode*. The *VirtualManager* controls each *StorageSlice* associated to each *VirtualNode* deployed inside a *PhysicalNode*. An instantiation of the forth level modules is depicted in Figure B.4.



(a) Storage (b) Virtual Node

Figure B.4: Storage and virtual node modules representation

Aiming at designing a very flexible and configurable simulation module, many parameters were defined to characterize the features of each one of the defined compound and simple modules. Based on those parameters it is possible to configure: (i) number of physical nodes, and inside each physical node it is also possible to define the number of virtual nodes and virtual pipes; (ii) physical and virtual network links features, such as bandwidth, delay, and error; (iii) the transmission rate, load of requests for each virtual application inside the virtual nodes; (iv) monitoring intervals and self-organizing evaluation cycles. The flexibility on the configuration of the environment enables that different scenarios of simulation could be created without the necessity of rewriting the Network Virtualization module.

B.2 Monitoring Process

The self-organization of the virtual elements inside the substrate network is only possible if accurate monitored information is provided for the decision-making mechanism. Therefore, an special monitoring process was developed for the self-organizing P2P approach. This process is composed of two phases. The first one is the acquisition of raw

information that is illustrated in Figure B.5. The second is the manipulation and storage of such raw information (depicted in Figure B.6) so that it can be later on handled by the *SelfOrganization* module.

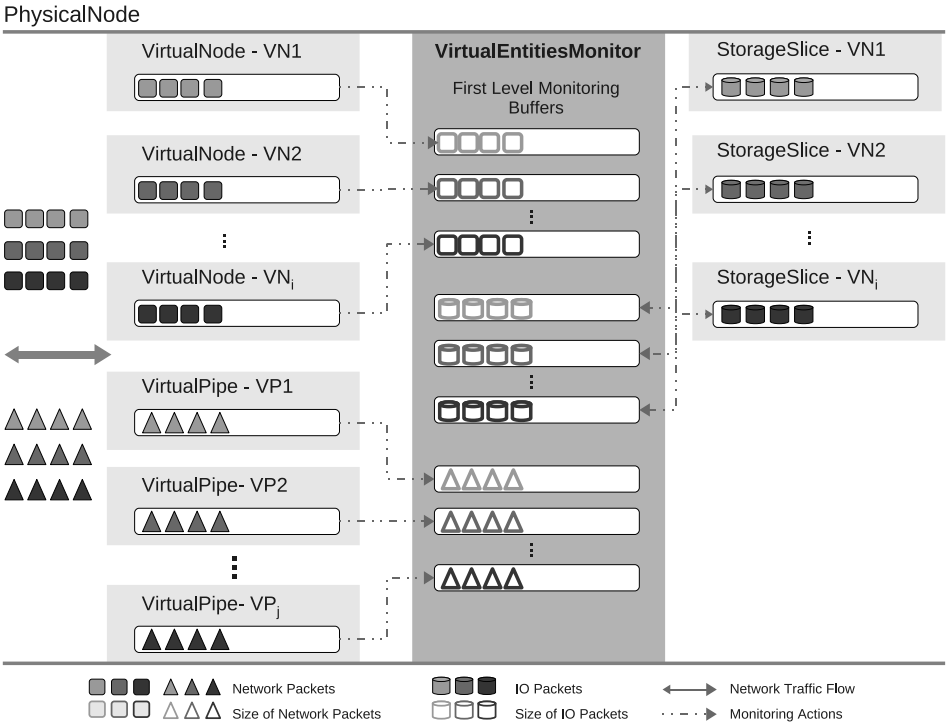


Figure B.5: Gathering monitoring information inside *PhysicalNode*

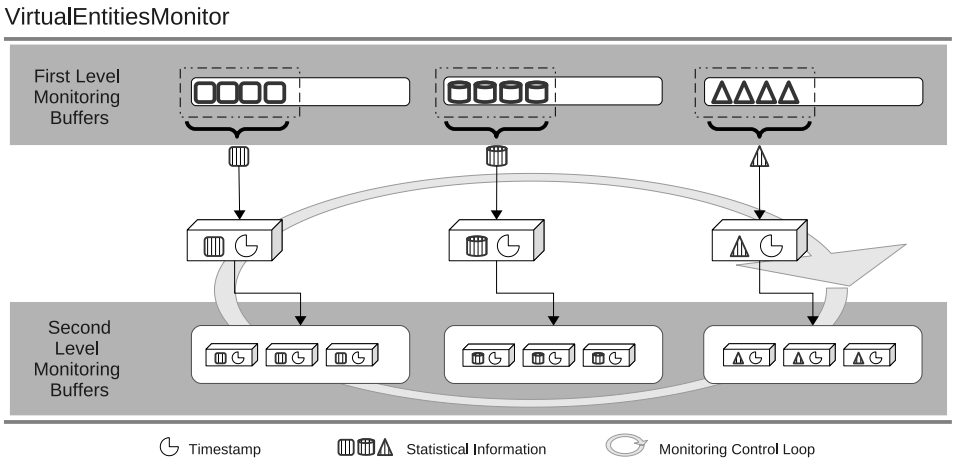


Figure B.6: Monitoring control loop inside *VirtualEntitiesMonitor* module

The raw information is actually the size of the network and IO packets that are transmitted/received by the virtual elements (virtual nodes or pipes and storage slices). In the substrate network provider perspective, the important information is the amount of data being used by the virtual slice and not which kind of information is inside each packet. Based on this perspective, the substrate network provider can also enable transparency

between the management of the substrate network and what is happening inside each virtualized network. The size of each arrived/sent packet (*i.e.* raw information) is stored in temporary buffers called *First Level Monitoring Buffers*, as illustrated in Figure B.5.

The acquisition of raw information is an asynchronous process that depends on the activities of the virtual slices being monitored inside the physical node. Therefore, there is not an specific monitoring cycle to gather the information of the *First Level Monitoring Buffers*. In contrast, Figure B.6 illustrates that the monitoring information inside the *Second Level Monitoring Buffers* is generated every monitoring cycle inside the *VirtualEntitiesMonitor* module. Statistical operations are applied to the raw information and the result is stored in the *Second Level Monitoring Buffers*. Finally, the *SelfOrganization* module uses this statistical information to evaluate the need of performing some reorganization of the virtual elements.

APPENDIX C - TÓPICOS INVESTIGADOS

Esse apêndice têm como objetivo apresentar de forma compacta os principais tópicos investigados nesse tese. Inicialmente, a motivação para realização desse trabalho será apresentada. Em seguida, os princípios da tese serão definidos e discutidos. A metodologia de execução da investigação deste trabalho, baseia-se no desenvolvimento de estudos de caso. Sendo assim, dois estudos de caso são examinados nessa tese e apresentados nesse apêndice. Para finalizar, as discussões finais, conclusões e trabalhos futuros são relatados.

C.1 Motivação

Gerenciamento de redes é uma disciplina importante cujo principal objectivo é a manutenção das infraestruturas de comunicação e o funcionamento correto de serviços de rede. Ao longo dos anos, vários desafios foram enfrentados pela comunidade de pesquisa em gerenciamento de redes. Exemplos de tais desafios são: definição de protocolos de gerenciamento de redes padronizados, *e.g.*, *Simple Network Management Protocol* (SNMP) (HARRINGTON; PRESUHN; WIJNEN, 2002), *Common Open Policy Service* (COPS) (PEREIRA; GRANVILLE, 2008), NETCONF (*Network Configuration*) (BHUSHAN; SCHÖNWÄLDER, 2009); investigação de formas de execução das operações de gerenciamento de redes, *e.g.*, centralizado e distribuído (MARTIN-FLATIN; ZNATY; HABAUX, 1999) (SCHÖNWÄLDER; QUITTEK; KAPPLER, 2000) (PAVLOU, 2007). Embora as pesquisas propostas até agora tenham introduzido melhoramentos no gerenciamento de redes, existem inúmeros novos desafios (*e.g.*, redução da intervenção humana, heterogeneidade, escalabilidade, confiabilidade) que não estão completamente cobertos pelas formas de gerenciamento até então propostas. A comunidade de pesquisa, dessa forma, começou a investigar novas alternativas de modelagem e desenvolvimento de soluções de gerenciamento de redes (SCHÖNWÄLDER et al., 2006).

Uma das alternativas investigadas é o uso de computação autonômica em gerenciamento de redes (SAMAAN; KARMOUCH, 2009). O termo Computação Autonômica (*Autonomic Computing - AC*) foi usado pela primeira vez em 2001 (HORN, 2001) e sua definição mais aceita foi apresentada por Kephart and Chess (KEPHART; CHESS, 2003). As principais ideias por trás da computação autonômica são a redução da intervenção humana e o aumento do comportamento autônomo dos sistemas (*e.g.*, auto-configuração, auto-monitoramento). Na visão da comunidade de gerenciamento de redes, existem também evidências do uso de termos como comunicações autonômicas, auto-gerenciamento, e propriedades *self-** para referenciar as ideais relacionadas com computação autonômica (SAMAAN; KARMOUCH, 2009) (STRASSNER et al., 2009) (BOUABENE et al., 2010) (MANZALINI; ZAMBONELLI, 2006). Outra alternativa explorada pela comunidade científica é o emprego de tecnologias Peer-to-Peer (P2P) nas soluções de gerenciamento

de redes. O uso de P2P tem sido reivindicado como sendo uma forte alternativa para o melhoramento da conectividade entre domínios administrativos heterogêneos, escalabilidade, confiabilidade, e cooperação entre os gerentes (GRANVILLE et al., 2005) (XU et al., 2008) (FIORESE; SIMÕES; BOAVIDA, 2009). De fato, a literatura mostra que as investigações sobre o uso de propriedades *self*-* e tecnologias P2P constituem as maiores correntes de alternativas atualmente seguidas pela comunidade de pesquisa em gerenciamento de redes.

Geralmente, as investigações relacionadas com propriedades *self*-* e computação autonômica aplicadas ao gerenciamento de redes focam na definição de modelos de alto nível (STRASSNER et al., 2008) (BERGLUND et al., 2008)(CHONG et al., 2009), tais como ontologias (SERRANO; SERRAT; STRASSNER, 2007) (KEENEY; LEWIS; SULLIVAN, 2007) (FUENTES; VERGARA; CASTELLS, 2006) e políticas (SIMMONS; LUTFIYYA, 2005) (MEER et al., 2006) (ZHAO; CHEN; CRESPI, 2008) (HADJIANTONIS; PAVLOU, 2008), que são capazes de conduzir as ações autônomas do sistema. Em contraste, pesquisas aplicando P2P ao gerenciamento de redes são basicamente focadas na definição de infraestruturas de comunicação da solução de gerenciamento (GRANVILLE et al., 2005) (BARSHAN; FATHY; YOUSEFI, 2009), infraestruturas de suporte à disseminação de informação (YALAGANDULA et al., 2006) (BINZENHÖFER et al., 2006) e à conectividade dos elementos de gerenciamento (*e.g.*, gerentes, agentes, dispositivos) (ZHOU; RENESSE, 2005) (FALLON et al., 2007). Sendo assim, analisando o caso das propriedades *self*-* vê-se o problema da falta de investimentos aproximando os modelos de alto nível às infraestruturas de gerenciamento, enquanto no caso de solução baseadas em P2P experiencia-se o problema oposto. Neste sentido, propriedades *self*-* e P2P constituem técnicas complementares, cujo uso conjunto dessas duas técnicas pode ser explorado como uma nova alternativa na modelagem e desenvolvimento de soluções de gerenciamento de redes.

Atualmente, existem pesquisas mostrando a viabilidade da utilização integrada de propriedades *self*-* e tecnologias P2P. Por exemplo, existem propostas utilizando propriedades *self*-* para prover um melhor gerenciamento da própria rede P2P (BEJAN; GHOSH, 2004) (SACHA et al., 2006) (JONES et al., 2009). Outras propostas, empregam propriedades *self*-* para melhorar o desempenho de sistemas P2P (NTARMOS; TRIANTAFILLOU, 2005) (BISKUPSKI; DOWLING; SACHA, 2007) (XIE; MIN; DAI, 2009). Entretanto, o emprego conjunto de propriedades *self*-* e P2P com o objetivo de construir soluções de gerenciamento de redes não foi, até agora, extensivamente investigado, e as potencialidades e desvantagens dessa união continuam incertas. Por esse motivo, essa tese visa introduzir novos conhecimentos e ideias envolvendo a integração de propriedades *self*-* e P2P no contexto do desenvolvimento de novas alternativas para a modelagem de soluções de gerenciamento. Para conduzir o processo de introdução de novos conhecimentos, esta tese estabelece uma linha de investigação baseada em uma hipótese e em perguntas fundamentais, que são apresentadas a seguir.

***Hipótese.* "A combinação de propriedades *self*-* e técnicas P2P habilita a modelagem e o desenvolvimento de aplicações de gerenciamento de redes verdadeiramente distribuídas e cooperativas."**

Perguntas fundamentais associadas à hipótese foram definidas, e são apresentadas abaixo, com o intuito de guiar as investigações dessa tese.

Pergunta fundamental I. *Quais são as características introduzidas pelas propriedades self-* e técnicas P2P na modelagem e execução de soluções de gerenciamento de redes?*

Pergunta fundamental II. *Quais são os benefícios do emprego de técnicas self-* P2P na construção de soluções de gerenciamento de redes em comparação às formas self-* centralizado ou self-* hierárquico?*

Pergunta fundamental III. *Quais os custos impostos pelo uso da forma self-* de gerenciamento de redes?*

A metodologia utilizada para executar as investigações é baseada no desenvolvimento de estudos de caso e na obtenção dos *issues* de integração da modelagem das propriedades *self-** e P2P para o gerenciamento de rede. Dois estudos de caso foram definidos. O primeiro investiga o uso da propriedade de auto-cura e P2P para o gerenciamento de falhas de plataformas de gerenciamento. O outro explora o uso integrado da propriedade de auto-organização e interações P2P com intuito de gerenciar a quantidade de tráfego em ambientes de rede virtualizados. Baseado na observação das características de cada uma das soluções desenvolvidas nos estudos de caso, foi possível derivar resultados relacionados com o uso integrado das propriedades *self-** e P2P para o gerenciamento de redes. Esses resultados são analisados de forma a responder as perguntas fundamentais dessa tese. Além disso, essas respostas também constituem as contribuições dessa tese para a área de gerenciamento de redes.

C.2 Princípios da Tese

A literatura mostra que o desenvolvimento de soluções de gerenciamento de redes têm sido focada em aspectos ligados a protocolos, arquiteturas, *frameworks*, e APIs - *i.e.* aspectos morfológicos - enquanto a modelagem das aplicações de gerenciamento têm sido negligenciada. Pressionados pelos desafios da área, a comunidade de gerenciamento de redes começou a empregar soluções mais sofisticadas, como por exemplo propriedades *self-** e técnicas P2P. Entretanto, a maior parte dessas soluções mais sofisticadas ainda é focada nos aspectos morfológicos do gerenciamento de redes. Em paralelo, discussões sobre a falta de soluções realmente distribuídas e cooperativas, além da falta de investimentos em algoritmos, começaram a ser enfatizadas pela comunidade de gerência de redes (SCHÖNWÄLDER et al., 2006), mas nenhuma solução foi proposta até agora.

Em vista de situação descrita acima, as investigações conduzidas nessa tese são direcionadas a explorar diferentes ângulos e aspectos daqueles empregados até então, *i.e.*, dos aspectos morfológicos. Nesta tese, é de interesse especial examinar como as aplicações de gerenciamento de redes podem ser modeladas quando técnicas sofisticadas como propriedades *self-** são combinadas com a natureza distribuída de tecnologias e técnicas P2P. O objetivo é investigar as capacidades de conectividade e cooperação de P2P como âncora para prover verdeira distribuição e cooperação na execução de aplicações de gerenciamento de redes que utilizam propriedades *self-**.

A quantidade de ambientes de redes que poderiam ser investigados nessa tese é muito grande. Entretanto, indiferentemente à natureza da rede observada (cabeado ou não cabeada), mudanças dinâmicas e a necessidade de redução da intervenção humana são algumas das características mais presentes nas redes atuais. Nesta tese, acredita-se que

a presença dessas características implica um conjunto específico de requisitos de gerenciamento, e esse conjunto justifica o emprego de aplicações de gerenciamento de redes baseadas em *self*-* P2P. Os **requisitos de gerenciamento** de redes considerados nessa tese são: *(i) uso eficiente dos recursos*; *(ii) ações gerenciamento ágeis*; *(iii) ações de gerenciamento devem ser transparentes para os usuários utilizando os recursos da rede*; *(iv) explorar mais comportamentos paralelos e simultâneos nas ações de gerenciamento*.

Com o intuito de desenvolver a alternativa *self*-* P2P para o gerenciamento de redes, tornou-se necessário a definição de ***issues* de integração** das propriedades *self*-* e P2P. Através da utilização dos *issues* de integração pode-se desenvolver modelos, arquiteturas, e algoritmos. Nesta tese definiram-se os seguintes *issues* de integração: *(i) gerentes devem ter uma imagem comum sobre a tarefa de gerenciamento*; *(ii) utilização de informações locais*; *(iii) utilização de algoritmos paralelos e distribuídos para o processo de decisão*; *(iv) criação de self-elements simples*; *(v) redução da coordenação explícita e global*. Tendo como base o emprego dos *issues* de integração, os estudos de caso foram modelados e desenvolvidos.

Nesta tese, dois estudos de caso foram examinados. O primeiro está relacionado com a investigação de auto-cura e P2P para o gerenciamento de falhas de sistemas de monitoramento. O segundo estudo de caso explora auto-cura e interações P2P a fim de gerenciar os recursos físicos em ambientes de redes virtualizadas. A descrição mais detalhada de cada estudo de caso é feita a seguir.

C.3 Estudo de Caso I - Auto-cura de Plataformas de Monitoramento

Monitoramento é essencial no gerenciamento de redes para possibilitar a identificação de problemas na infraestrutura de comunicação das organizações modernas. Entretanto, os sistemas de monitoramento atuais não são capazes de recuperar seus próprios elementos que falharam, forçando o administrador da rede a recuperar manualmente o sistema de monitoramento ocasionalmente em falha. Este estudo de caso, por esse motivo, ataca esse problema através da introdução da propriedade auto-cura na solução de monitoramento. A proposta desse estudo de caso combina a disponibilidade e transparência de comunicação de *overlays* P2P com a propriedade de auto-cura.

A modelagem e a avaliação da solução de auto-cura P2P foram aplicadas para a construção de um sistema de monitoramento de Controle de Acesso a Rede (*Network Access Control* - NAC). A solução é capaz de prover a propriedade de auto-cura através da divisão de processos distintos: as funções de detecção de falhas e recuperação do sistema. A detecção de falhas é executada dentro dos serviços de gerenciamento que monitoram dispositivos finais, enquanto a recuperação é realizada por serviços especiais chamados *self-healing* (o qual decide quando novas instâncias de serviços em falha devem ser ativadas) e *self-configuration* (que é responsável por ativar as novas instâncias dos serviços como reação das decisões tomadas pelo serviço *self-healing*).

Através da análise dos resultados experimentais, pode-se concluir que o número de instâncias dos serviços *self-healing* e *self-configuration* não influencia tanto a performance do sistema. Esses resultados também permitem afirmar que as falhas simultâneas de serviços de gerenciamento não influenciam expressivamente a performance. Sendo assim, um administrador de redes que deseje empregar a solução de monitoramento com auto-cura não deve concentrar seus esforços em encontrar um número ideal de instâncias de *self-healing* e *self-configuration*. Os experimentos utilizaram 2 e 4 instâncias, respectivamente, e o sistema respondeu satisfatoriamente. O fato que deve ser observado,

entretanto, é a solução de comunicação em grupo disponível na rede gerenciada: multicast permite uma recuperação mais rápida com um consumo menor de tráfego de rede. Infelizmente, multicast IP nem sempre pode ser provido, e unicast acaba sendo a solução escolhida nessas circunstâncias.

Os aspectos mais importantes que devem ser observados na solução de auto-cura são o número de *peers* utilizados no *overlay* de gerenciamento P2P, e o número de instâncias dos serviços. Com um número pequeno de instâncias, não existe necessidade para utilização de muitos *peers*. Entretanto, com um grande número de instâncias, o número de *peers* deve aumentar consistentemente, caso contrário, na ocorrência de falhas o tempo de recuperação irá aumentar e mais largura de banda será consumida em vista do intenso tráfego P2P gerado. Além das considerações relacionadas com a performance da solução de auto-cura P2P, existem importantes aspectos ligados à modelagem do comportamento cooperativo dessa solução. A solução proposta apresenta uma postura de cooperação intencional, e para atingir tal postura utiliza os seguintes métodos de cooperação: uso de grupos, comunicação, especialização, colaboração através do compartilhamento de informações, e coordenação de tarefas.

C.4 Estudo de Caso II - Auto-organização de Recursos Virtuais

De acordo com pesquisas recentes, virtualização é um técnica promissora para o desenvolvimento de redes futuras (CHOWDHURY; BOUTABA, 2009) (NIEBERT et al., 2008)(FEAMSTER; GAO; REXFORD, 2007)(BERL et al., 2008). A ideia chave está na identificação e separação de dois papeis: o provedor do substrato que possui e mantêm a rede física, e o provedor virtual que constrói sua infraestrutura através do aluguel de fatias de recursos do provedor do substrato. Considerando o provedor virtual como um entidade que vende serviços, a vantagem da virtualização está no fato de que os custos de executar uma infraestrutura física podem ser *outsourced* para um provedor externo. Um desafio importante em virtualização de redes é o gerenciamento dos recursos do substrato da rede. Técnicas de gerenciamento sofisticadas devem ser utilizadas em tal gerenciamento. Algumas dessas técnicas oferecidas por alternativas de auto-gerenciamento destacam-se como apropriadas para resolução dos desafios de gerenciar eficientemente o uso dos recursos do substrato. Esta tese, desta forma, mostra como a integração da propriedade de auto-organização e técnicas P2P podem ser utilizadas no gerenciamento dos recursos da rede do substrato.

Na solução proposta, ao invés de utilizar somente as capacidades de conectividade das infraestruturas de P2P e manter a forma de gerenciamento da rede baseada em hierarquias e delegação, são fortemente utilizados modelos de interações P2P cooperativas para realização da tarefa de auto-organização. O modelo proposto está fundamentado na identificação de elementos gerenciáveis com perspectivas complementares. A união da propriedade de auto-organização com iterações P2P resulta em uma alternativa de auto-organização P2P capaz de reorganizar os elementos da rede durante tempo de execução. Essa alternativa pode ser empregada se as seguintes condições forem atendidas: (i) os elementos que precisam ser movidos estão gerando/recebendo tráfego de rede; (ii) existe um tráfego de *cut-through* dentro da rede; (iii) existe uma abstração entre a rede de mais baixo nível e a infraestrutura de mais alto nível onde os elementos que devem ser movidos estão executando; e (iv) é possível identificar perspectivas complementares para um mesmo evento que possa estar acontecendo nos elementos envolvidos na reorganização.

De fato, as perspectivas complementares são a principal peça na modelagem da solução de auto-organização P2P, porque elas criam as condições para o desenvolvimento de um postura cooperativa. A ideia de compartilhar a responsabilidade da tomada de decisão (entre as entidades que querem mover seus recursos e as que desejam receber tais recursos) forma a base da construção dos seguintes métodos de cooperação: uso de grupos, comunicação, colaboração através do compartilhamento de informações e decisões, e a coordenação de tarefas. Complementando as considerações sobre a modelagem, os experimentos mostraram que os benefícios da utilização da solução de auto-organização P2P são mais expressivos em termos de redução da carga de tráfego do que quando comparados com a latência dos pacotes. Na realidade, a alta latência observada durante o processo de reorganização é um importante aspecto que deve ser observado, pois ela pode se tornar uma questão que previna a utilização da solução proposta em diferentes cenários.

C.5 Discussão dos Principais Resultados

A alternativa *self-** P2P foi concebida para ser uma solução de gerenciamento de redes e serviços completamente distribuída e cooperativa. A base de tal alternativa está no fato de que ela explora o desenvolvimento de aplicações de gerenciamento mais sofisticadas, que utilizam características paralelas, distribuídas e concorrentes dos ambientes em que está inserida. Por essa razão, os *issues* de integração agem como um guia de como se deve atingir o desenvolvimento das aplicações sofisticadas desejadas. Esta tese mostrou que existe um casamento entre os atributos de concorrência definidos por Zomaya (ZOMAYA, 1996) e os *issues* de integração estabelecidos para a modelagem de soluções *self-** P2P. Esse casamento prova a forte tendência da alternativa *self-** P2P em empregar mecanismos paralelos, distribuídos, concorrentes, e cooperativos desde a fase de concepção das aplicações de gerenciamento.

A análise dos *issues* de integração mostrou que existe uma clara distinção entre as estratégias empregadas em cada estudo de caso. O primeiro - *i.e.*, auto-cura P2P - utiliza uma estratégia mais distribuída para resolução do problema de recuperação do sistema. Grupos e *self-elements* com papéis distintos são orquestrados a fim de prover a propriedade de auto-cura. Diferentemente, a solução de auto-organização P2P possui uma estratégia mais individualizada com a interação baseada em pequenos grupos formados por vizinhos para prover a propriedade *self-**. Embora as estratégias sejam muito diferentes em sua forma, o objectivo final da modelagem e desenvolvimento de soluções de gerenciamento completamente distribuídas e cooperativas foi atingida em ambas estratégias. A comparação isolada dos métodos de cooperação utilizados pelas soluções de auto-cura P2P e auto-organização P2P, poderia levar a conclusão de que as duas soluções são muito similares. A partir dessa conclusão, poderia-se pensar que uma mesma estratégia de obtenção de comportamento cooperativo foi adotada por ambas soluções. Entretanto, como provado pela discussão anterior, as estratégias e formas de modelagem das soluções são completamente diferentes em sua essência, e uma inconsistência surge a partir dessas duas análises.

Essa inconsistência entre o que pode ser esperado dos métodos de cooperação e as formas evidenciadas de modelagem da cooperação levam a conclusão de que existem relacionamentos implícitos entre os *issues* de integração, as características para prover a propriedade *self-**, e os métodos de cooperação. O ponto inicial para efetuar uma análise mais detalhada desses relacionamentos está no fato de que ambos tipos de propriedades *self-** podem ser modeladas com quase os mesmo métodos de cooperação, entretanto, as

características para prover a propriedade *self*-* são completamente diferentes.

A conclusão retirada dos fatos citados acima é que existem diferentes níveis de cooperação associados com as alternativas *self*-* P2P propostas nesta tese. Além disso, esses níveis estão intimamente relacionados com o tipo da natureza da propriedade *self*-*, *i.e.*, se elas podem ser divididas em múltiplos *self-elements* ou se são indivisíveis e providas por um único tipo de *self-element*. Com base no raciocínio descrito, foi possível identificar duas dimensões associadas com alternativas *self*-* P2P: a natureza da propriedade *self*-* e os níveis de cooperação. A dimensão na natureza das propriedades *self*-* é composta pelas categorias divisível e indivisível, enquanto a dimensão dos níveis de cooperação é composta pelas categorias elemento, tarefa, e sistema.

O processo de modelagem de uma propriedade *self*-* pode começar pela definição de qual o nível de cooperação será empregado ou por qual tipo de natureza da propriedade *self*-* deseja ser utilizada. No momento em que uma das dimensões é fixada a outra será uma consequência direta da primeira. Além disso, a definição da modelagem de propriedades *self*-* implica indiretamente em quais tipos de capacidades P2P podem ser exploradas. É importante ressaltar que capacidades P2P relacionadas com infraestrutura podem ser exploradas por quaisquer modelagens das propriedades. Entretanto, essa afirmação tende a não ser válida para capacidades P2P relacionadas com aplicações.

C.6 Conclusões e Trabalhos Futuros

A investigação conduzida nessa tese mostra a união dos conceitos de alto nível de propriedades *self*-* com capacidades P2P tanto de infraestrutura como de aplicação. Para guiar essa investigação, uma hipótese foi definida e a metodologia empregada para verificar essa hipótese baseia-se na definição de requisitos de gerenciamento, *issues* de integração para a modelagem conjunta de propriedades *self*-* e P2P, e a identificação e desenvolvimento de dois estudos de caso. O primeiro explora a modelagem de auto-cura P2P para a manutenção de plataformas de gerenciamento de redes, enquanto o segundo investiga a auto-organização P2P para o gerenciamento de recursos em ambientes de redes virtualizadas. Os resultados experimentais mostram que o uso integrado de propriedades *self*-* e P2P minimizam a intervenção humana e melhoram a performance das tarefas de gerenciamento de redes. As principais contribuições dessa tese são listadas abaixo.

- A alternativa *self*-* P2P possibilita a definição de um comportamento completamente distribuído e verdadeiramente cooperativo nas soluções de gerenciamento de redes.

- A alternativa *self*-* P2P muda o ângulo de desenvolvimento das soluções de gerenciamento de redes dos aspectos morfológicos para os relacionados com os algoritmos. O foco passa ser explorar comportamentos paralelos e cooperativos dos *peers* de gerenciamento que executam os algoritmos de gerenciamento.

- O gerenciamento pode ser embutido dentro dos elementos gerenciados sem a necessidade de gerentes externos. Dessa forma, existe um suporte melhor para execução simultânea e paralela de ações de gerenciamento.

- As duas contribuições do primeiro estudo de caso. Primeiramente, a definição de uma plataforma de gerenciamento capaz de recuperar instancias de gerenciamento sem a intervenção humana. Segundo, a determinação de existência de *trade-offs* entre o tempo de recuperação e o tráfego de rede associado.

- As contribuições do segundo estudo de caso são: definição de uma arquitetura de gerenciamento distribuída para virtualização de redes, utilizando para tanto aplicações de gerenciamento de redes baseadas em perspectivas complementares e interações P2P; introdução de interações entre *peers* vizinhos representa uma quebra de paradigma das soluções até então desenvolvidas no campo de engenharia de tráfego; definição de heurísticas para identificação de tráfego *cut-through* na rede de substrato utilizando somente informação local sem inspeção de pacotes.

Nesta tese, também foram identificadas outras oportunidades de pesquisa a partir da investigação conduzida. Dois exemplos principais dessas oportunidades de trabalhos futuros são: (i) investigar quais são as consequências da modelagem de múltiplas e diferentes propriedades *self*-* utilizando a alternativa *self*-* P2P para o mesmo cenário de gerenciamento de redes; (ii) identificar maneiras de formalizar os conceitos introduzidos pela alternativa *self*-* P2P para definição de aplicações de gerenciamento de redes distribuídas e cooperativas, de forma que o método formal possa ser aplicado ao desenvolvimento de diferentes tipos de propriedades *self*-*. Para concluir, através do que foi dito anteriormente, é possível constatar que o trabalho desenvolvido nessa tese mostra possibilidades de repensar a forma como gerenciamento de redes é modelado e executado.

APPENDIX D - SCIENTIFIC PRODUCTION

This appendix presents the scientific production achieved during the development of the thesis described in this document. The papers published within the time of this PhD can be divided into two categories: the ones directly related to the objectives of the thesis, and the ones that contributed to create the knowledge necessary to gain maturity and evolve the ideas of the thesis. The list of these papers is presented as follows[2].

1. Cristiano Bonato Both, CLARISSA CASSALES MARQUEZAN, Rafael Kunst, Jéferson Campos Nobre, Juergen Rochol, Lisandro Zambenedetti Granville. Self-adapting Mobile WiMAX networks: Rethinking the Design of Connection Admission Control System. **ACM SIGCOMM Computer Communication Review**.

   - *Status.* Indication to revise and resubmit (December 2010).
   - *Paper category in relation to the Thesis.* Contributed to the knowledge.
   - *Description.* This paper presents the initial ideas of a self-adapting design of an Connection Admission Control (CAC) for WiMAX networks. Here, a Point-Multi-Point WiMAX network is considered, therefore a centralized self-adapting mechanism was proposed. The next step of this research is to consider a distributed WiMAX environment (*e.g.*, mesh or relay), where self-adapting P2P strategies are required.

2. CLARISSA CASSALES MARQUEZAN, Giorgio Nunzi, Marcus Brunner, Lisandro Zambenedetti Granville. Distributed Autonomic Resource Management for Network Virtualization. **Proceedings of 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010)**, 19-23 April 2010, Osaka, Japan

   - *Status.* Approved and waiting to be published.
   - *Paper category in relation to the Thesis.* Directly related.
   - *Description.* This paper presents details of the self-organizing P2P approach, such as the heuristics, the distributed algorithm, and the evaluation scenario.

[2]It is not presented in this appendix the papers that were not accepted to be published.

3. CLARISSA CASSALES MARQUEZAN, Jéferson Nobre, Lisandro Zambenedetti Granville, Giorgio Nunzi, Dominique Dudkowski, Marcus Brunner. Distributed Reallocation Scheme for Virtual Network Resources. **Proceedings of IEEE International Conference on Communications(ICC 2009)**, 14-18 June 2009, Dresden, Germany, ISBN: 978-1-4244-3435-0, pp. 1-5.

    - *Status.* Approved and published.
    - *Paper category in relation to the Thesis.* Directly related.
    - *Description.*This paper presents the initial ideas of the self-organization P2P approach. The focus of this paper is the motivation why the network virtualization needs a self-organizing model in order to manage the physiccal resources.

4. CLARISSA CASSALES MARQUEZAN, André Panisson, Lisandro Zambenedetti Granville, Giorgio Nunzi, Marcus Brunner. Maintenance of Monitoring Systems Throughout Self-Healing Mechanisms. **Proceedings of the 19th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2008)**, 22-26 September 2008, Samos Island, Greece, ISBN: 978-3-540-85999-4, pp. 176-188

    - *Status.* Approved and published.
    - *Paper category in relation to the Thesis.* Directly related.
    - *Description.* This paper presents the self-healing P2P approach applied to the case study of a P2P-based network monitoring overlay.

5. CLARISSA CASSALES MARQUEZAN, Carlos Raniery Paula dos Santos, Jéferson Campos Nobre, Maria Janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco, Lisandro Zambenedetti Granville. Self-managed services over a P2P-based Network Management Overlay. **Proceedings of the 2nd Latin American Autonomic Computing Symposium (LAACS 2007)**, 12-13 September 2007, Petrópolis, Brazil

    - *Status.* Approved and published.
    - *Paper category in relation to the Thesis.* Directly related.
    - *Description.* This paper brings the first insights related to the self-management elements that later influenced the definition of the self-healing P2P approach.

6. CLARISSA CASSALES MARQUEZAN, Carlos Raniery Paula dos Santos, Ewerton Monteiro Salvador, Maria Janilce Bosquiroli Almeida, Sérgio Luis Cechin, Lisandro Zambenedetti Granville. Performance Evaluation of Notifications in a Web Services and P2P-Based Network Management Overlay. **Proceedings of the 31st IEEE International Computer Software and Applications Conference (COMPSAC 2007)**, 23-27 July 2007, Beijing, China, ISBN: 0-7695-2870-8, pp. 241-250

- *Status.* Approved and published.
- *Paper category in relation to the Thesis.* Contributed to the knowledge.
- *Description.* This paper presents the first experiences of working with network management overlays, that was very useful during the definition of the self-healing P2P approach.

7. Ricardo Lemos Vianna, CLARISSA CASSALES MARQUEZAN, Everton Polina, Leandro Bertholdo, Maria Janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco. An Evaluation of Service Composition Technologies Applied to Network Management. **Proceedings of the 2007 IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)**, 21-25 May 2007, Munich, Germany, ISBN: 1-4244-0799-0, pp. 420-428

   - *Status.* Approved and published.
   - *Paper category in relation to the Thesis.* Contributed to the knowledge.
   - *Description.* The work in this paper provided the possibility to establish an important partnership that was used to during the development of the case study related to the self-healing P2P approach.

Self-adapting Mobile WiMAX networks: Rethinking the Design of Connection Admission Control System

Cristiano B. Both
cbboth@inf.ufrgs.br

Clarissa C. Marquezan
clarissa@inf.ufrgs.br

Rafael Kunst
rkunst@inf.ufrgs.br

Jéferson C. Nobre
jcnobre@inf.ufrgs.br

Juergen Rochol
juergen@inf.ufrgs.br

Lisandro Z. Granville
granville@inf.ufrgs.br

Institute of Informatics - Federal University of Rio Grande do Sul
Porto Alegre, RS, Brazil

ABSTRACT

WiMAX is a connection-oriented wireless network that provides guaranteed QoS requirements. One importante component in WiMAX QoS provisioning is the Connection Admission Control (CAC). In order to analyze the admission of a connection, CAC systems must be aware of the characteristics of the network. We define such characteristics as vertical and horizontal aspects of WiMAX networks. Current researches focus on handling vertical aspects, and neglected the horizontal ones. In this paper, we introduce a self-adapting CAC design able to handle both aspects. Our design incorporates self-* properties in order to autonomously select the most appropriate CAC algorithm, according to changes on the characteristics of the network. We present the architecture of the self-adapting CAC system; two case studies that can benefit from the proposed design; and, finally, the advantages and challenges of this proposal.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design

General Terms

Design, Management

Keywords

Connection Adminission Control, Mobile WiMAX, Self-* Properties, Network Management

1. INTRODUCTION

Worldwide Interoperability for Microwave Access (WiMAX) has been considered a cost-effective alternative for metropolitan broadband Internet access when wired solutions are not geographically or economically viable [1]. WiMAX follows the specifications of the IEEE 802.16 standard [2] and, in practice, both are referred as synonyms. IEEE 802.16 defines five classes of services [2] as part of its strategy to support Quality of Service (QoS). One key component in WiMAX QoS provisioning is the Connection Admission Control (CAC), which is responsible for preventing a WiMAX network of becoming overloaded by managing the admission of new connections. In fact, IEEE 802.16 assumes the existence of CAC but it does not define how it should be actually designed or implemented, which opens research opportunities in both academia and industry.

CAC solutions employ, inside each WiMAX base station, an algorithm that decides whether new connections, requested by subscriber stations, can be admitted or not. Very simple CAC algorithms take this decision based on the occupation of the bandwidth allocated to each WiMAX class of service [3]; as long as a class of service can accommodate the traffic of new connections, these new connections are admitted. Recent investigations, however, take a step further and propose more sophisticated CAC algorithms whose decisions take into account also the dynamics of network traffic requirements (*e.g.*, maximum latency and tolerated jitter) and wireless physical conditions (*e.g.*, signal power and channel bandwidth). In this paper, we call these the *vertical aspects* that guide the decisions of a CAC solution.

Considering the current state-of-the-art, there are several approaches to address the vertical aspects of a CAC decision [3] [4] [5] [6]. For example, a statistical CAC mechanism was proposed by Yu *et. al.* [7] where the CAC decision considers two classes of services and channel bandwidth. Bashar and Ding [8] describe an adaptive and cross-layer CAC solution, where the decision is based on two classes of services, Signal-to-Noise Ratio (SNR), channel bandwidth, and signal power. Ghazalt *et. al.* [9] present the design of a probabilistic and self-configuring CAC algorithm that uses non conventional fuzzy-based admission control method for downlink connections and considers two classes of services plus channel bandwidth to make the decision.

Observing the proposals of CAC systems above described, it is possible to verify that they are designed having in mind specific sets of network features. Indeed, these sets of features are associated to the expected demands generated by the users, and these demands create a network usage profile. In this sense, each one of the aforementioned proposals are associated to one specific network usage profile. However, if the predominant demands of the users change, the current CAC proposals are not able to change their behavior in favor of a more suitable profile. In this paper, we refer as the *horizontal aspects* of a CAC solution those that are related to different profiles. Thus, it is clear to notice that today's CAC solutions properly address the vertical aspects, while partially or fully neglecting the horizontal ones.

Aiming at properly addressing both vertical and horizontal aspects, we propose to rethink the design of traditional WiMAX CAC. In this article we present a novel CAC design

where properties widely refereed as self-* [10] are incorporated into the CAC architecture in order to autonomously select the most appropriate CAC algorithm according to the predominant network usage profile. Our proposal introduces in traditional CAC architectures three components: (a) a repository of CAC algorithms from where the most appropriated one can be chosen; (b) a knowledge base that stores scheduler feedbacks, traffic requirements, and physical conditions; and (c) one algorithm devoted to adapt the CAC system to different network usage profiles. Therefore, the major contributions of this work are: (i) it addresses WiMAX dynamics and heterogeneity beyond traffic requirements and physical conditions, and (ii) it supports changes on WiMAX networks without manual intervention.

The remaining of this paper is organized as follows. In Section 2 we provide the context of this research where the WiMAX and CAC are reviewed. The self-adapting CAC design proposed in this paper is introduced in Section 3. Two case studies that can benefit from the proposed design are described in Section 4. The advantages and challenges of this proposal are described in Section 5. Finally, Section 6 depicts the conclusion of this work.

2. BACKGROUND

WiMAX is a connection-oriented wireless network, *i.e.*, each connection must be first admitted by the base station [2]. In order to admit a traffic flow, WiMAX base stations must receive information from the fixed and/or mobile subscriber stations requesting a connection (hereafter, subscriber is the term used for both requester stations). The IEEE 802.16 standard defines ranging and register messages (Figure 1) to provide such information to the CAC system.

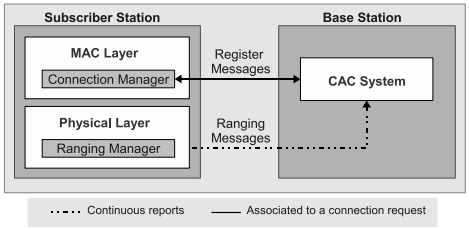


Figure 1: WiMAX messages related to CAC System

As presented in Figure 1, ranging and register messages are exchanged between the subscriber station (through the ranging manager and the connection manager) and the base station. Details about ranging and register messages, information provided to CAC systems, and the characteristcs of such systems are depicted as follows.

2.1 WiMAX Messages for CAC Systems

A ranging message is exchanged continuosly between the subscriber station and the base station. This type of message is not associated to a connection request, but to the physical conditions of the wireless channel used by a subscriber station. Ranging messages are divided in two phases: initial and periodic ranging. The goal of the initial phase is the negotiation of transmission power and time synchronization between the subscriber and the base station. In this phase the subscriber station receives a basic Connection IDentification (CID). The periodic ranging phase, in its turn, starts right after the initial negotiation between subscriber and base station. In this phase the subscriber receives a primary CID to exchange two types of messages: Ranging Request (RNG-REQ) and Ranging Response (RNG-RSP). These messages are used to handle time variant conditions of the wireless channel between subscriber and base station. Examples of information inside these messages are signal power, modulation scheme, and coding rate. In addition, based on the content of these messages it is possible to derive other physical information. For example, Signal to Noise plus Interference Ratio (SNIR) is calculated based on the signal power information inside the ranging message.

On the other hand, register messages are directly related to requests to admit a new connection. In order to establish a new connection, the subscriber station needs to request a registration in the base station. The registration is granted after exchanging two register messages: Register Request (REG-REQ) and Register Response (REG-RSP). The register messages enclose traffic descriptors that contain information, such as QoS requirements, associated to the traffic flow of the new connection. Moreover, these messages can assume three types of actions to manage the QoS requirements of the connections: Dynamic Service Addition (DSA), Dynamic Se rvice Change (DSC), and Dynamic Service Deletion (DSD). Based on these actions, the subscriber station can request modifications on its QoS requirements to the base station in order to handle changes on traffic conditions.

2.2 Providing information to CAC System

As presented above, inside ranging and register messages there is information that can be used by the CAC system. Indeed, this information can be used as input parameters for the decision-making process of admitting or not a connection. Table 1 shows some information commonly retrieved from the physical and traffic descriptors of the ranging and register messages used as parameters in the CAC system.

Table 1: Partial View of Potential Parameters

Type	Parameters
Physical Descriptor	Power Level
	Available Bandwidth
	Modulation and Coding
	Handoff
	Location Update
	Signal to Noise plus Interference Ratio
Traffic Descriptor	Minimum Reserved Traffic Rate
	Maximum Sustained Traffic Rate
	Maximum Reserved Traffic Rate
	Maximum Traffic Burst
	Tolerated Jitter
	Maximum Latency
	Class of Service

The most important parameter, retrieved from the register message, is the class of service. The CAC system analyzes all other traffic parameters according to the class of service informed by the subscriber station. Today, WiMAX networks support five classes of services: Unsolicited Grant

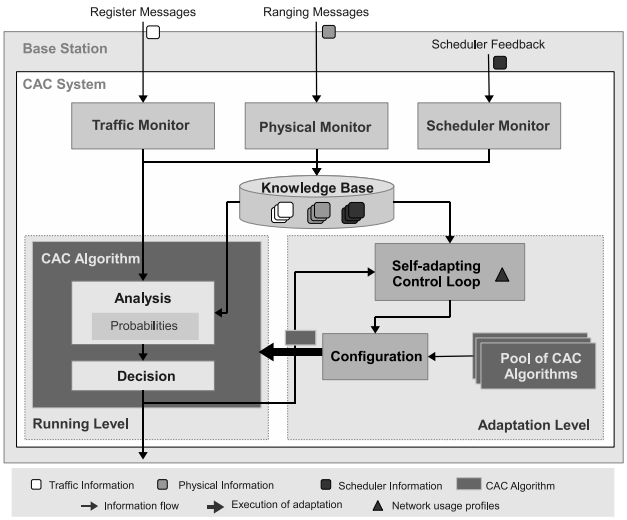


Figure 2: Architecture of Self-adapting CAC Design

Services (UGS), extended real-time Polling Service (ertPS), real-time Polling Service (rtPS), non-real time Polling Service (nrtPS), and Best Effort (BE). Indeed, these classes of services reflect the heterogeneity of network applications.

Current CAC researches use different combination of parameters. For example, Lee and Kim [11] uses UGS and rtPS classes of services, considering minimum reserved traffic rate for each class, modulation and coding scheme, and SNIR. On the other hand, Qin *et. al.* [12] employ only one class of service (rtPS) but they consider more physical and traffic parameters than the previous proposal (minimum reserved traffic rate, maximum reserved traffic rate, power level, SNIR, available bandwidth, handoff, and modulation and coding scheme). Actually, the parameters used in a CAC system describe its characteristics.

2.3 Characteristics of CAC Systems

The core of a CAC system is the algorithm that analyzes the parameters, calculates probabilities (*e.g.*, Connection Blocking Probability (CBP), Connection Dropping Probability (CDP), and Handoff Failure Probability (HFP)), and decides whether a connection should be admitted or not. The type and number of parameters, and probabilities used by the algorithm are related to the objectives on running the CAC system.

According to Ahmed [13] the major objectives on using a CAC system could be classified into four groups: (i) guarantee parameters related to QoS; (ii) manage the network revenue; (iii) prioritize some services/classes of service; and (iv) provide fair resource sharing. In fact, an objective for using a CAC system is related to the definition of a network usage profile. Currently, CAC systems for WiMAX networks typically support the execution of one or very few network usage profiles, which transforms a CAC system into a static solution that might be able to address vertical aspects but that fails on properly handling horizontal aspects.

3. PROPOSAL

The main objective of the self-adapting CAC design is to handle vertical and horizontal aspects associated to WiMAX networks. To achieve this objective, the features of the network environment (physical conditions and traffic requirements) must be constantly monitored by the base station. Whenever necessary the CAC system must be adapted to enhance the connection admission analysis on behalf of the predominant features. The architecture designed to support execution of CAC decisions, identification of network features, and adaptation are depicted in Figure 2.

The architecture of the self-adapting CAC design is composed of: three monitors, a *Knowledge Base*, and two parallel levels of execution. The *Traffic Monitor* and *Physical Monitor* receive messages from the subscriber stations. However, the nature of such information is different, once the former receives register messages (associated to connection requests) and the later receives ranging messages (associated to conditions of the wireless channel). The *Scheduler Monitor* receives feedbacks from the scheduler system inside the base station. This system is responsible for allocating the resources (*i.e.* downlink and uplink bandwidth) for a connection admitted by the CAC system. All information received by these monitors is stored for a certain period of time in the *Knowledge Base*, and is later used in different ways by both *Running* and *Adaptation* execution levels.

3.1 Running Level

In the self-adapting CAC architecture, the *Running Level* is responsible executing a *CAC Algorithm* that considers

information related to traffic requirements, physical, and scheduler conditions to admit or not a connection. The *CAC Algorithm* is composed of *Analysis* and *Decision* elements. The *Analysis* element calculates the *Probabilities* related to connection admission (*e.g.* CBP, CDP, HFP) and sends them to the *Decision* element that evaluate the admission or not of a connection.

The information used by the *Analysis* element is retrieved from both *Traffic Monitor* and *Knowledge Base*. Whenever a request to connect (*i.e.*, a register message) arrives at *Traffic Monitor* it is forwarded to the *Analysis* element of the *CAC algorithm*. Once this element has the traffic requirements of the incoming request, it retrieves physical information and the status of the bandwidth resources from the *Knowledge Base*. The physical information retrieved is specifically related to the subscriber station requesting the connection. The *Analysis* element uses the primary CID of the subscriber station inside of the received register message to search the *Knowledge Base* and get the last ranging information associated to the same primary CID. It is possible to have an accurate information about physical conditions of each subscribe station on the same coverage of the base station and also about the consumed bandwidth, because the ranging messages and the scheduler feedbacks are stored at the *Knowledge Base*.

We believe that the *Running Level* encourages the development of optimized CAC algorithms to manage the changes on vertical aspects of the network. Meanwhile, the optimization of solutions to handle the changes on the horizontal aspects are delegated to the *Adaptation Level*.

3.2 Adaptation Level

The components of the *Adaptation Level* are presented in Figure 3. The *Collector*, *Predominant Detector*, *Evaluation*, and *Configuration* are components designed to execute the adaptation analysis and enforcement of changes. External (*Knowledge Base* and feedback from the *Decision* element of the CAC algorithm) and internal (*Pool of Network Usage Profiles* and *Pool of CAC Algorithms*) information are used by those components to support their processes.

In this work, *network usage profile* is the term used to characterize a set of the network features associated to expected demands generated by users. Most of current CAC proposals for WiMAX networks consider that a network usage profile is defined by a class of service. However, for the self-adapting CAC design, the network usage profile can be defined according to classes of services, or traffic requirements, or physical conditions, or scheduler conditions, or by the combination of any of these CAC parameters. Examples of network usage profiles are shown in Table 2.

The *rtPS_Profile* is an example of profile associated to real time applications (*e.g.*, video conference), while *Handoff_Profile* considers all connections that are moving between two base stations regardless the class of service of the traffic. We defined that for each network usage profile there must be an associated CAC algorithm that considers the parameters specified on this profile. Thus, a network usage profile can be supported by a base station only if it is stored at the *Pool of Network Usage Profile* and its associated CAC algorithm is stored on the *Pool of CAC Algorithms*.

Based on the network usage profile, the *Collector* component starts the adaptation analysis. Indeed, the *Collector* continuously monitors and processes information retrieved

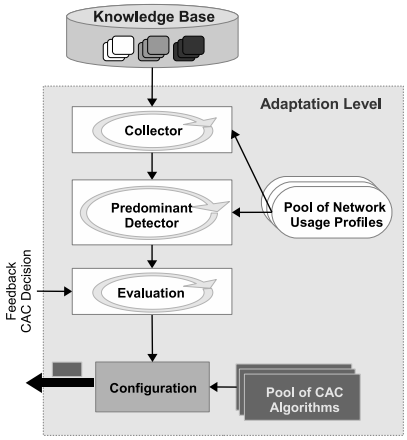


Figure 3: Internal Architecture of Adaptation Level

Table 2: Examples of Profiles

Network Usage Profile 1	
Name	rtPS_Profile
Class of Service	rtPS
Traffic Requirements	Minimum Reserved Traffic Rate
Physical Conditions	Modulation and Coding
	SNIR
Scheduler Conditions	Available Downlink and Uplink Bandwidth
Network Usage Profile 2	
Name	Handoff_Profile
Class of Service	UGS, ertPS, rtPS, nrtPS, BE
Traffic Requirements	Minimum Reserved Traffic Rate
	Maximum Latency
Physical Conditions	Power Level
	Handoff
Scheduler Conditions	Available Bandwidth

from the *Knowledge Base* following the parameters used to characterize a network usage profile. During each cycle, statistical information about each parameter associated to the network usage profiles are generated and stored in the "collector buffer". At the end of the cycle, the analyzed data inside the *Knowledge Base* must be erased mainly to free space of the buffers associated to traffic requirements, physical conditions, and scheduler feedbacks.

The "collector buffer" is then analyzed by the *Predominant Detector* component to determine which is the predominant profile so far. This determination is conducted by matching the statistical information of the "collector buffer" with the different types of network usage profiles supported by the base station. At the end of this process, the resultant predominant profile is stored in a "detector buffer".

The *Evaluation* component is responsible for analyzing the content of the "detector buffer". To determine whether an adaptation is required, the *Evaluation* component has to go through four stages. First, it extracts statistical information from the "detector buffer" in order to determine which is the predominant network usage profile. Second, it is necessary to identify whether the chosen profile is different from the current one. If the current and the new network usage profiles are different, than an adaptation should be executed. In the third stage, the *Evaluation* component checks the reliability of the chosen predominant profile by looking for evidences on the feedbacks arrived from CAC decisions. Finally, the costs involved to execute the adaptation are evaluated, and in the end, the adaptation is performed only if there is a favorable cost-effective tradeoff.

The *Configuration* component is the one responsible for enforcing the new predominant network usage profile. Thus, the *Running Level* is prepared to change the CAC algorithm in execution by the one associated to the new predominant network usage profile. The new algorithm will be retrieved from the *Pool of CAC Algorithms*. The interruption of admitting or not connections must be avoided, and, in this sense, the exchange process between the running and the new CAC algorithms is of great importance for the overall performance of the self-adapting CAC system.

In next section we present some case studies in which the self-adapting CAC design can be applied in order to improve the network QoS. We present scenarios that describe ordinary situations observed in current WiMAX networks.

4. CASE STUDIES

In the following subsections we describe two case studies. One considering vertical aspects of WiMAX networks, and the other handling horizontal aspects. For each case study, we show how the self-adapting CAC design can be employed.

4.1 From Fixed to Mobile Mode

In the first case study, imagine a worker listening to an on-line radio station using for this an WiMAX device connected to the Internet. In a first moment, the worker is at home, thus, for the WiMAX base station "A", his/her device is operating as a fixed subscriber station. Then, in a second moment, the worker moves from home to the office and keeps listening to the on-line radio station using the same WiMAX device. Consider also that the office of this worker is physically placed in the same coverage area of the WiMAX base station "A". In this case, during the movement from home to the office, the WiMAX device changes its operation mode from fixed to mobile subscriber in the perspective of the base station "A". Finally, when the worker arrives at the office, he/she keeps being served by the same base station, and the WiMAX device assumes again a fixed operation mode.

Now, consider that not only one worker is behaving as described in the scenario above, but that several workers are behaving like this within the same coverage area of the WiMAX base station "A". In this case, some workers might be requesting new connections in a mobile mode, others will be finishing their connections on fixed mode and new ones are initiated, and the connections of those workers that were initiated in the fixed mode must be kept. Moreover, this scenario of movement usually occurs twice every working day (*i.e.*, at the time when most workers go to their offices, and when they return to their homes). Thus, if the network usage profile employed during these periods of movement does not consider important parameters to identify the changes on the operation mode of the subscribers (*e.g.*, signal power, location update, and SNIR), than these dynamic changes will be neglected. The consequence of ignoring these changes will be the degradation of the QoS requirements of the connections being established during the transition periods.

Fortunately, the self-adapting CAC design can identify this kind of situation and exchange the network usage profile on behalf of the new physical conditions of the network. The components of the *Adaptation Level* are able to match the changes on the network conditions with the network usage profiles supported by the base station. Thus, if there is a profile that takes into account the subscriber modes of operation, than there is a CAC algorithm that is able to analyze those important parameters for this context. In this way, the CAC algorithm can be exchanged every time the workers move from home to the office, and vice-versa.

The scenario discussed in this case study is related to the *vertical* aspects of WiMAX networks, where there is no change on the network demands of the users, (*i.e.* the users keep listening to the on-line radio station) but the changes on physical conditions impact the decisions of admitting new connections. It is not easy to define a single CAC algorithm able to analyze the very different and dynamic types of traffic and physical parameters. Therefore, the self-adapting CAC design offers a better support to handle vertical changes on WiMAX networks.

4.2 From Handoff to Service Priorization

In the second case study, we present an example of how horizontal aspects of WiMAX networks can be treated by the self-adapting CAC design. For this purpose, imagine a base station that is physically placed nearby a village and a highway. Consider that this base station is configured with a CAC algorithm that prioritizes handoff pattern, *i.e.*, requests of subscriber stations exchanging from one base station to another. Due to the placement of this base station, it is reasonable to think that requests of the handoff pattern should be served first, because such requests potentially belong to mobile stations on the highway, and not to the subscriber stations on the village. Now, consider that the inhabitants of the village want to watch the matches of the National League using their WiMAX connections. This means that, probably, during more or less two hours the base station should receive more multimedia connections requests (*i.e.*, rtPS class of service) than requests of handoff pattern.

Considering the scenario of this case a study, it is very probable that the CAC algorithm prioritizing handoff admits any kind of traffic asking for a handoff pattern (*e.g.*, BE traffic like HTTP of the users at the highway) regardless the class of service of new requests. For example, new multimedia requests to connect from users in the village (with QoS requirements) would be dropped or blocked. This situation reveals an unfair use of WiMAX network resources, specially if the multimedia has become the predominant traffic.

Despite the fact that there might exist intersections among the parameters used by a CAC algorithm to prioritize the admission of handoff pattern and another for multimedia priorization, the essence of the decision is different. Such essence is related to the relevant probabilities for each CAC algorithm. Thus, for this scenario, it is not possible to think of solely changing the parameters used by the CAC algo-

rithm. Indeed, the network usage profile has changed and the current CAC algorithm needs to be replaced for another one able to admit connections according to the new profile of the network. In this case, the self-adapting CAC design appears as a proper model to solve the unfair use of the network presented in this case study.

In fact, both case studies discussed in this section are typically faced by administrators of WiMAX networks. The proposed self-adapting CAC design can be used in order to detect changes in the predominant network profile and consequently to choose the CAC algorithm that best fits to the new network usage profile. Although it improves the network QoS, there are some challenges that must be faced by the implementation of such a CAC design.

5. ADVANTAGES AND CHALLENGES

The proposed self-adapting CAC design has advantages but also introduces challenges that should be better investigated. Thus, this section presents these aspects in details.

The first significant advantage of our proposal is to address WiMAX dynamics and heterogeneity beyond traffic requirements and physical conditions, by using a network usage profile. Indeed, this advantage creates new opportunities of researches, once it is necessary to define which network usage profiles are relevant to improve QoS requirements on WiMAX networks.

The second advantage is to support changes on CAC algorithm without manual intervention. Currently, the CAC algorithms are normally configured only one time in the factory, and whether changes are required they are executed manually by the network administrator through firmware update. Nevertheless, the self-adapting CAC design dismisses this manual intervention due to the fact that it is able to on-line exchange CAC algorithms.

Moreover, our proposal opens some challenges that must be investigated in order to guarantee that a self-adapting CAC design can bring more benefits than drawbacks. The first challenge is to define which are the characteristics of a predominant profile, and how to calculate such predominance based on the match of the records inside the *Knowledge Base* and network usage profiles supported by the base station. For example, employing thresholds would be an initial attempt to determine a predominant profile.

A second challenge is to identify when is the appropriate moment to change the running profile for another one that represents the new conditions of the network. One alternative to help on identifying the changing moment is to evaluate the costs to perform the change. In fact, this alternative opens news challenges because it is also necessary to identify which are the relevant information to be considered for the evaluation of such costs. Avoiding instabilities on the system is also a challenge. For example, the self-adapting CAC system should have a protection mechanism to avoid successive changes in CAC algorithm that might degenerate the overall performance of the system.

Another challenge that should be investigated is related to the possibilities of implementing the system. The *Running* and *Adaptation* levels must run in parallel, therefore, these levels may be implemented in different processors. For example, the CAC algorithm inside the *Running Level* must operate with a very low response time, thus being deployed in a dedicated processor. However, this algorithm can be exchanged during the operation of the CAC system. In this case, a re-programmable processor would be a better solution for deploying the *Running Level*. On other other hand, the *Adaptation Level* does not require a fast response time, so it can be implemented on a general purpose processor.

Finally, we believe that during the implementation of a self-adapting CAC design several other challenges will appear and will contribute for the better understanding and development of QoS support in WiMAX networks.

6. CONCLUSION

The self-adapting CAC design proposed in this paper is able to adapt to both vertical and horizontal aspects of WiMAX networks. Two major factors enhance the capabilities of adaptation of this proposal. The first one is the introduction of a *Knowledge Base* that temporary stores the information that impacts the CAC decisions. Second is the definition of network usage profiles to characterize the network utilization beyond the simplistic classes of service. Based on the match of supported network usage profiles with the information in the *Knowledge Base*, the self-adapting CAC system can determine which is the predominant profile and change the CAC algorithm in execution for another more suitable for the conditions of the network.

The proposed approach creates new opportunities of researches. For example, studies about which are the basic network usage profiles that must be supported by a base station; how it is possible to deploy the *Running* and *Adaptation* levels; determine the relevant information for defining the costs of adapting the system; etc. Finally, as future work, we will define the function that describes the predominance of a network usage profile, and provide an implementation of the architecture presented in this paper.

7. ACKNOWLEDGEMENT

This work was partly funded by the CNPq Brazilian Research Agency.

8. REFERENCES

[1] Y. A. Sekercioglu, M. Ivanovich, and A. Yegin, "A survey of MAC based QoS implementations for WiMAX networks," *Computer Networks*, vol. 53, no. 14, pp. 2517–2536, 2009.

[2] "IEEE standard for local and metropolitan area networks - part 16: Air interface for fixed and mobile broadband wireless access systems - IEEE std. 802.16-2005," December 2005.

[3] B. Zhu, K. Xue, H. Lu, and P. Hong, "Fair connection admission control scheme for IEEE 802.16e systems," vol. 1, no. 1. IEEE, October 2008, pp. 1–4.

[4] B. Rong, Y. Qian, K. Lu, H.-H. Chen, and M. Guizani, "Call admission control optimization in WiMAX networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 4, pp. 2509–2522, 2008.

[5] B. Rong, Y. Qian, and K. Lu, "Integrated downlink resource management for multiservice WiMAX networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6.

[6] B. Rong, Y. Qian, and H.-H. Chen, "Adaptive power allocation and call admission control in multiservice WiMAX access networks," *IEEE Wireless Communications*, vol. 14, no. 1, pp. 14–19, February 2007.

[7] K. Yu, X. Wang, S. Sun, L. Zhang, and X. Wu, "A statistical connection admission control mechanism for multiservice IEEE 802.16 network," in *Proceedings of IEEE 69th Vehicular Technology Conference*, April 2009, pp. 1–5.

[8] S. Bashar and Z. Ding, "Admission control and resource allocation in a heterogeneous OFDMA wireless network," *IEEE Transactions on Wireless Communications*, vol. 8, no. 8, pp. 4200–4210, August 2009.

[9] S. Ghazalt, Y. H. Aout, J. B. Othman, and F. Nait-Abdesselam, "Applying a self-configuring admission control algorithm in a new QoS architecture for IEEE 802.16 networks," in *Proceedings of IEEE Symposium on Computers and Communications*, July 2008, pp. 1023–1028.

[10] N. Samaan and A. Karmouch, "Towards autonomic network management: an analysis of current and future research directions," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 22–36, August 2009.

[11] J. Y. Lee and K. B. Kim, "Statistical connection admission control for mobile WiMAX systems," in *Proceedings of IEEE Wireless Communications and Networking Conference*, April 2008, pp. 2003–2008.

[12] C. Qin, G. Yu, Z. Zhang, H. Jia, and A. Huang, "Power reservation-based admission control scheme for IEEE 802.16e OFDMA systems," in *Proceedings of IEEE Wireless Communications and Networking Conference*, March 2007, pp. 1831–1835.

[13] M. H. Ahmed, "Call admission control in wireless networks: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 1, pp. 49–68, 2005.

Distributed Autonomic Resource Management for Network Virtualization

Clarissa C. Marquezan and Lisandro Z. Granville
Institute of Informatics - UFRGS
Porto Alegre, RS, Brazil
Email: {clarissa,granville}@inf.ufrgs.br

Giorgio Nunzi and Marcus Brunner
NEC Europe Network Laboratories
Heidelberg, Germany
Email: {nunzi,brunner}@nw.neclab.eu

***Abstract*—Network virtualization is an emerging trend claimed to reduce the costs of future networks. The key strategy in network virtualization is of slicing physical resources (links, routers, servers, etc.) to create virtual networks composed of subsets of these slices. One important challenge on network virtualization is the resource management of the physical or substrate networks. Sophisticated management techniques should be used to accomplish such management. The sophisticated techniques offered by autonomic communications rise as an appropriated alternative to address the challenges of managing the efficient use of substrate resources on network virtualization. Thus, this paper proposes a distributed self-organizing model to manage the substrate network resources. An evaluation scenario is depicted and simulations show that approximately 36.8% of the network traffic load can be spared when the self-organizing model is enabled in the evaluated scenario.**

I. Introduction

Autonomic communications is a suitable approach to deal with complex and dynamic networks. The key concept behind autonomic communications is of building sophisticated networks capable of managing themselves in order to deal with changes from the surrounding environment. Among the researches employing autonomic communications, the ones related to virtual technologies deserve special attention due to their complexity, dynamics, and potential to be economically exploited. Network virtualization is an example of virtual technology that is emerging as a promising cost-effective solution for future network deployments [1].

Network virtualization differs from current virtual machine and virtual network approaches. The difference relies on the type of resources that are virtualized. Virtual machine employs multiplexing techniques to virtualize CPU, memory, storages, and device interfaces [2]. Virtual networks multiplex physical links and build paths connecting edge customers [3]. In this case, the network elements connecting the edge customers (*e.g.*, routers, switches, etc.) are not perceived as elements of the customers network, but they form a "tunnel" connecting edges. Finally, network virtualization multiplexes all substrate network resources (*i.e.*, physical links, routers, servers, base stations, etc.) [4]. The resultant environment is a set of slices of substrate resources that forms an entire new network deployed on top of a physical infrastructure. Indeed, this set forms a virtual network capable of running its own protocols, routing process, services, and management solutions.

One of the major benefits of network virtualization is to outsource the operational costs associated to physical infrastructures to a single provider. For example, multimedia providers may deploy their services, like IPTV services, without dealing with high investments on the physical infrastructure [5]. As a complement, a physical or substrate provider could multiplex its physical network to enforce multiples multimedia providers. The resource management of a traditional physical networks demands a lot of effort. So, it is reasonable to think that resource management on network virtualization requires even more efforts in order to be successfully accomplished. For instance, consider the problem where two distinct virtual networks require conflicting amount of resources in the same substrate network area. If this problem were detected during the deployment phase, the substrate provider could employ traditional techniques, like traffic matrix optimization and load balancing, and achieve a successful usage of the substrate resources after the deployment of the virtual networks. However, if this problem occurs during the lifetime of the virtual networks it becomes necessary to make dynamic and on-line changes on the environment. In this case, the employment of traditional techniques present limitations because in general they use a centralized, total-view, and off-line approaches to manage the network resources [3]. The major limitations are low responsiveness to network changes, overhead introduced by the management traffic related to the central entity, and high latency of analysis and enforcement of changes.

According to the aforementioned, we believe that more sophisticated management techniques are demanded in order to cope with changes on the amount of substrate resources used by the virtual networks during their lifetime. The techniques offered by autonomic communications, like self-awareness, self-configuration, and self-organizing, rise as an appropriated alternative to address the challenges of maintaining the efficient use of substrate resources on network virtualization. Thus, this paper proposes a distributed self-organizing model to manage the substrate network resources. This model is based on parallel and distributed algorithms running inside each substrate node, and thus dismissing any kind of central entity. Based on local information, the substrate node decides to self-organize the substrate network in order to cope with the changes on traffic loads of the virtual networks. The self-organizing purpose is to reduce the overall traffic load of the

substrate network by moving virtual nodes.

The main contributions of this paper are: the definition of a distributed management architecture for network virtualization and the definition of a self-organizing model to maintain an efficient resource usage of the substrate network. To validate the proposed model, a scenario composed of virtual IPTV networks is simulated and evaluated in terms of traffic load and packet latency when the self-organizing model is enabled and disabled. The results show that approximately 36.8% of traffic load can be spared when our self-organizing model is employed in this scenario.

The remainder of this paper is described as follows. Section II brings the concepts used in this work. The self-organizing model is presented in Section III. The evaluation scenario and the associated results are, respectively, discussed on Section IV and V. The related work is presented in Section VI. Finally, Section VII brings the conclusions and future works.

II. Concepts

The high level concepts of the network virtualization model of our research have been previously described [6]. Now, we present the details of the modules and components of the substrate node architecture and concepts that enable the self-organization of network virtualization environment. Fig. 1 depicts the substrate node architecture that is composed of three modules: substrate resources, virtual manager, and virtual elements (virtual nodes and virtual pipes). Fig. 2 helps to illustrate the concepts that motivates the self-organizing model.

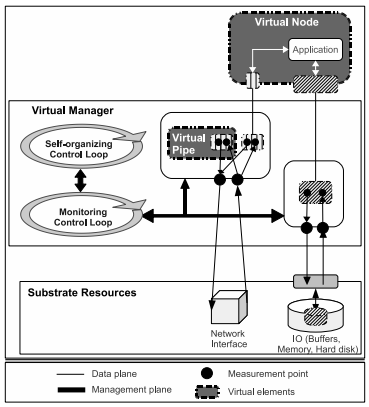


Fig. 1. Substrate node architecture

The substrate resources module comprises the physical resources of the substrate node, like network interfaces, IO (Input/Output) system (*e.g.*, buffers, memory, hard disk, etc.), CPU, etc. The virtual manager behaves as a middleware between the physical resources and the resources that are attributed to the virtual elements. The virtual manager also contains the components of the self-organizing model and one of the virtual elements (the virtual pipe). The components of the self-organizing model are: self-organizing and monitoring control loops, and local measurement points.

Virtual element modules are the virtual nodes and the virtual pipes. There is a conceptual difference between these two virtual elements related to what is physically deployed at the substrate network and what is perceived by the virtual network. An example that helps on clarifying this difference is presented in Fig. 2, where two virtual networks are deployed on top of a substrate network. More specifically, Fig. 2 depicts the virtual network topologies and the architecture of the substrate nodes supporting these virtual topologies.

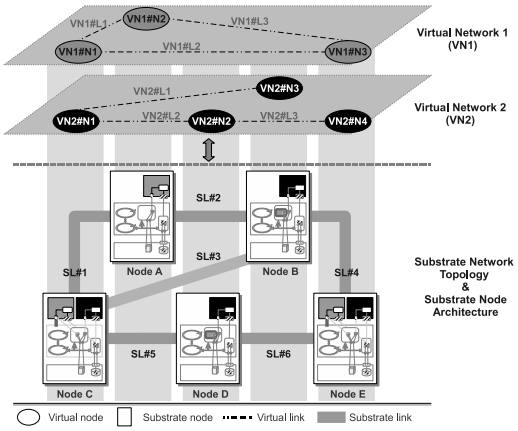


Fig. 2. Substrate and virtual networks perspectives

The substrate network in Fig. 2 is composed of five substrate nodes (“A”, “B”, “C”, “D”, “E”), and six substrate links (“SL#1”, “SL#2”, “SL#3”, “SL#4”, “SL#5”, “SL#6”). The Virtual Network 1 (VN1) is composed of three virtual nodes, “VN1#N1”, “VN1#N2”, and “VN1#N3”, being each one respectively mapped to the substrate nodes “C”, “A”, and “E”. The second virtual network, Virtual Network 2 (VN2), is composed of 4 virtual nodes, “VN2#N1”, “VN2#N2”, “VN2#N3”, and “VN2#N4”, respectively mapped to the substrate nodes “C”, “D”, “B”, and “E”. A virtual node corresponds to a set of physical resources from the substrate node that are wrapped together and assigned to a virtual network.

Another virtual element is also deployed in the architecture of the substrate nodes “B” and “D”, but it is not illustrated in VN1 topology. This element is the virtual pipe. While the virtual node is designed to belong to both virtual network topology and substrate node architecture, the virtual pipe is designed to be part solely of the substrate node architecture and transparent for the virtual network topology.

Indeed, a virtual pipe is a “connection” between two non physically adjacent virtual nodes. As observed in Fig. 2, not always two virtual nodes are allocated in adjacent sub-

strate nodes. For example, the virtual nodes of VN2 are all logically and physically adjacent, *i.e.*, logically due to the virtual topology, and physically because they are deployed in adjacent substrate nodes. However, the tuples of virtual nodes <VN1#N1,VN1#N3> and <VN2#N1,VN1#N3> of VN1 are only logically adjacent, since the virtual links "VN1#L2" and "VN1#L3" connecting the virtual neighbors are mapped to more than one substrate links. For example, "VN1#L2" is actually mapped to substrate links "SL#5" and "SL#6", while "VN1#L3" comprises substrate links "SL#2" and "SL#4".

In our view, there is an economical reason that encourages the use of both virtual pipes and virtual nodes. The deployment of a virtual node requires the assignment of more substrate resources than the deployment of a virtual pipe, because virtual pipes require resources solely for "tunneling" traffic and no other complex infrastructure (like applications, network stacks, etc.). The assignment of substrate resources means that costs will be charged to enable the use of these resources. Thus, the higher is the number of virtual nodes, the higher are the total costs to deploy a virtual network. Moreover, depending on the geographical demands, it is not cost-effective to deploy two physically adjacent virtual nodes, but somehow the traffic of one virtual node must arrive in its logically adjacent neighbor. In this sense, virtual pipes represent a less expensive connection between logically adjacent virtual nodes.

Inside a virtual pipe runs essentially cut-through traffic, *i.e.*, a traffic that is not originated inside the substrate node itself, but it just passes through the network interfaces of the substrate node. This means that for each virtual pipe associated to the traffic of a virtual network there is at least two substrate links that are being used but actually could be spared if the logically adjacent virtual nodes were also physically adjacent. Thus, aiming to spar substrate link resources we propose a self-organizing model that reduces the amount of cut-through traffic inside the substrate network by moving virtual nodes.

III. Self-organizing Model

The self-organizing model is based on a parallel and distributed algorithm that uses local information during the decision-making. This algorithm is executed by the self-organizing control loop inside the virtual manager module of each substrate node, as illustrated in Fig. 1. The local information is retrieved from the measurement points by the monitoring control loop. Afterwards, the self-organizing and the monitoring control loops exchange the measured information in order to verify whether a re-organization of virtual elements is required. Such re-organization is triggered by the detection of an overloaded substrate link and the identification of cut-through traffic inside this overloaded link.

A cut-through traffic can be identified under two perspectives: the virtual pipe where the traffic is passing by and the virtual node that is the source generating this traffic. Analysis of traffic under the virtual pipe perspective is called **receiving candidate perspective**, while analysis considering virtual nodes is called **moving candidate perspective**. Considering the fact that inside a substrate node might exist virtual nodes and virtual pipes, it becomes necessary to execute both perspectives inside the same control loop. In this way, the cut-through traffic associated either to virtual nodes or virtual pipes can be properly identified and self-organizing actions activated. An example of the high level execution of the self-organizing algorithm is presented in Fig. 3. This figure illustrates the stages of the self-organizing algorithm under the perspective of receiving and moving candidates. The self-organizing control loops depicted in Fig. 3 are part of the substrate nodes "B" and "E" previously presented in Fig. 2.

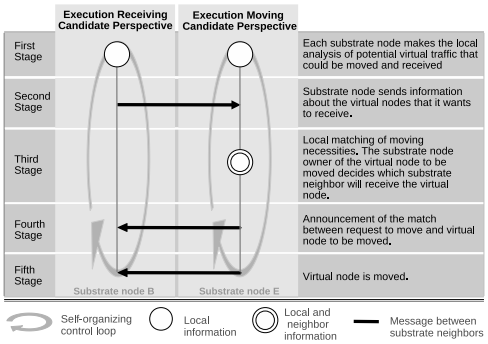


Fig. 3. Self-organizing algorithm

Five stages characterize the self-organizing algorithm. On the first stage, the capacity of the substrate links associated to a substrate node are locally analyzed by both perspectives of the self-organizing algorithm. The analysis starts with the identification of overloaded substrate links. Then, the traffic associated to each one of these overloaded links is investigated on the light of receiving and moving candidate perspectives. Heuristics are used to identify the cut-through traffic associated to each perspective. So far, the output of first stage is a list of virtual nodes that should be received and moved by the substrate node executing the self-organizing algorithm.

On the second stage, if the list of receiving and moving candidates is not empty, the receiving candidate perspective adopts an active behavior while the other perspective adopts a passive behavior. The receiving candidate perspective sends a message to its substrate neighbor requesting to receive a virtual node it supposes that is deployed on such neighbor. Meanwhile, the moving candidate perspective waits for a message from a substrate neighbor requesting to receive a virtual node belonging to its moving candidate list.

The third stage is characterized by the decision of moving a virtual node. Whenever the substrate node executing the moving candidate perspective receives the request message, it verifies if there is a match between the virtual node requested and its internal list of virtual nodes to be moved. Whether there is a successful match, the moving candidate perspective calculates the costs of re-organizing the virtual elements.

If there is a favorable cost-effect relationship on this re-organization, the request to move the virtual node is accepted.

The forth stage is the announcement of the decision to move a virtual node and the reservation of resources to host such virtual element at the substrate node that had its request accepted. Finally, at the fifth stage, the virtual node is moved and during the moving process all packets associated to this moving virtual node are buffered to be replayed after the end of the moving process. An example of the execution of the self-organizing algorithm is presented in Fig. 4, that is a partial view of the network showed in Fig. 2.

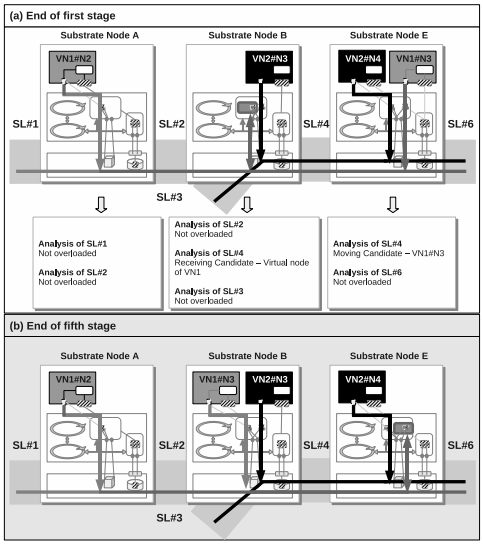


Fig. 4. Execution of the self-organizing algorithm

The internal state of the self-organizing algorithm at the end of the first stage is depicted in Fig. 4(a). The links that are not considered overloaded ("SL#1", "SL#2", "SL#3", and "SL#6") are discarded from the analysis of the self-organizing algorithm, while the overloaded one "SL#4" has its traffic investigated. When substrate node "B" analyzes the substrate link "SL#4", it discovers that this link is overloaded by the incoming traffic associated to the virtual pipe of VN1. Based on this, the self-organizing algorithm on substrate node "B" declares itself as a receiving candidate for a virtual node of VN1. During the first stage there is no communication between neighbors, and thus the self-organizing algorithm on substrate node "B" supposes that a virtual node of VN1 is deployed at substrate node "E" and then proceed to the next stages.

In parallel, the substrate node "E" also analyzes and declares "SL#4" as overloaded link, and considers the outgoing traffic associated to "VN1#N3" as the traffic overloading this substrate link. In this case, the self-organizing algorithm on substrate node "E" considers "VN1#N3" as a moving candidate, and waits for a request to move a virtual node from VN1 coming from "SL#4". When this request arrives in substrate node "E", there is a match between the virtual node that is requested to be moved (a virtual node of VN1) and the virtual node that should be moved from the substrate node "E" ("VN1#N3"). After this, the self-organizing algorithm inside the substrate nodes keeps executing until the last stage. The resulting environment is presented in Fig. 4(b), where "VN1#N3" has been moved to the substrate node "B" and the traffic from "VN1#N3" to "VN1#N2", that was considered the cut-through traffic overloading "SL#4", is not there any more.

A. Identification of an Overloaded Substrate Link

An overloaded link is the element that starts the evaluation of a possible re-organization of virtual resources on the substrate network. So, whenever the use on a substrate link sl overpass the threshold α, the next step is to identify which virtual link inside the overloaded substrate link is using the major amount of bandwidth. The output of this step is the list of virtual links overloading the substrate link sl_j ($OvVLink\_sl_j$ list), where j is the index of the sl connected to a substrate node. In the sequence, starts the process to determine whether the flow associated to the virtual link matches a cut-through traffic pattern. To determine this kind of match, two heuristics were defined and presented below.

B. Receiving Candidate Heuristic

The receiving candidate heuristic identifies a cut-through traffic when a virtual pipe is associated with the overloaded virtual link inside sl_j. This heuristic is based on the comparison of the incoming traffic against the outgoing traffic of a virtual pipe. Two conditions must be followed in order to declare the substrate node hosting the virtual pipe as a receiving candidate for a virtual node. First, the incoming traffic of the virtual pipe must be predominant on vl_{kj}, that is the virtual link vl of the virtual network k inside the substrate link sl_j. Second, the incoming traffic on vl_{kj} must be forwarded to one or multiple virtual links of the same virtual network k on the same substrate node. Every vl_{kj} belonging to $OvVLink\_sl_j$ is analyzed under these conditions.

C. Moving Candidate Heuristic

The moving candidate heuristic verifies if a virtual node is generating the outgoing traffic on vl_{kj}. Thus, we compare the outgoing traffic against the incoming traffic of each virtual link vl_{kj} inside the *OvVLink_*sl_j list. We assume that a virtual node uses not only virtual link resources to originate a traffic flow in vl_{kj}, but it also uses IO capacity. For example, before transmitting packets, the application inside the virtual node needs to buffer these packets and for doing this it needs to use the slices of IO resources assigned to itself. The moving candidate heuristic identifies a relationship between the outgoing traffic of link vl_{kj} with the amount of IO used by the virtual network (IOv_k) inside the substrate node. After this analysis, the list of moving candidates is created to be used during the second stage of the self-organizing algorithm.

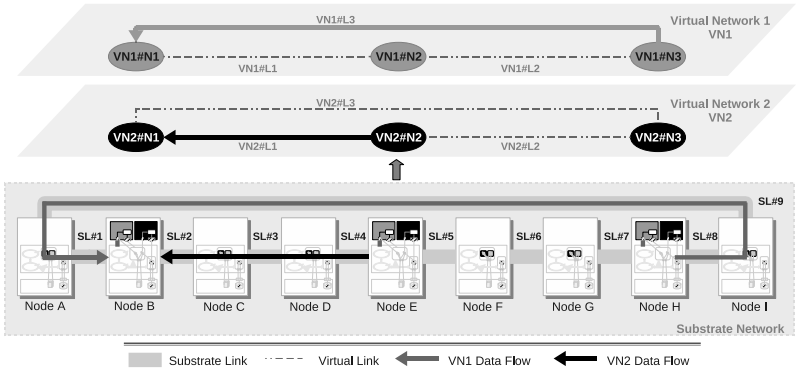


Fig. 5. Evaluation scenario

D. Implementation

To validate the substrate node architecture, concepts, and the self-organizing model proposed in this paper, we have to choose a substrate network model to be used. Despite the potential to gain a large market-share, network virtualization imposes large enhancements on the current network models, like new devices, transmissions management, etc. Given the problems on current models, we decided to work with a clean slate approach. Thus, we developed a network virtualization module based on Omnet++ simulator that uses a packet-oriented transmission mechanism with traffic-shaping.

One important aspect of the implementation is the monitoring process, since the self-organizing algorithm depends on this process to make decisions. A two-stage monitoring process was defined. The first stage is always active and the size of data passing through the measurement points is the monitored information buffered. The second stage is periodically activated and for the experiments the interval of 1.5 minutes is used. The monitored information on the first-stage buffer is summed and stored in a second-stage buffer. The self-organizing control loop uses the information of the second stage to determine the average amount of resources consumed within two self-organizing cycles. A sliding window keeps part of the information of the second-stage and the data from the first-stage is always erased. The sliding window helps to avoid that punctual high loads trigger constant reorganizations and lead the substrate network to an unstable state.

IV. Evaluation Scenario

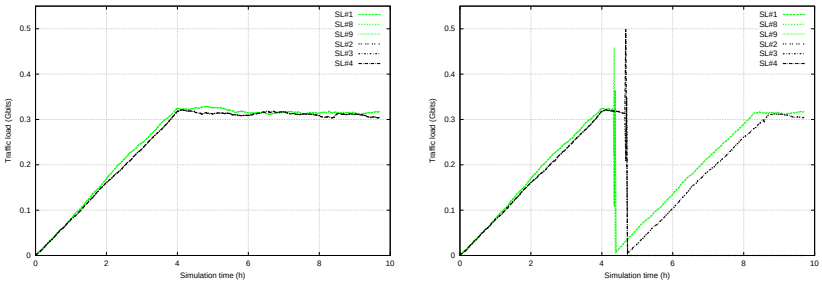
Network topologies and the initial mapping of the virtual IPTV networks are depicted in Fig. 5. The substrate network is composed of 9 substrate nodes and each virtual IPTV network is composed of 3 virtual nodes (physically separated by 2 virtual pipes). Each virtual node in Fig. 5 is an IPTV Video Hub Office (VHO) [5] able to store and transmit movies.

Two sets of flows are running inside the substrate network. The first one is associated to user's requests of the Virtual Network 1 (VN1). For VN1, the users connected to the VHO "VN1#N1" (inside node "B") are requesting movies that are associated to "VN1#N3" (inside node "H"). The movies are transmitted over the virtual link "VN1#L3", which is mapped to three substrate links "SL#1", "SL#8", and "SL#9". As depicted in Fig. 5, the flows of VN1 start at node "H" and arrive at the node "B". The second set of flows is associated to the Virtual Network 2 (VN2). The users connected to the VHO "VN2#N1" (inside node "B") are requesting movies that are stored at "VN2#N2" (inside node "E"). The transmission occurs through the virtual link "VN2#L1", which is mapped to substrate links "SL#2", "SL#3", and "SL#4". The flows of VN2 are originated at node "E" and arrive at node "B".

The bandwidth of each virtual link is 500 Mbits while the bandwidth of each substrate link is 1 Gbits. The size of the virtual storage associated to the VHOs of the virtual nodes is 50 GB, and the storage capacity of each substrate node is 100 GB. The size of the packets to be transmitted in the substrate links is fixed to 1 MB. The threshold to identify an overloaded link is the equivalent to 60% of the virtual link bandwidth. The amount of traffic inside virtual networks is mainly influenced by the number of movies requested by the users of the virtual networks. For this scenario, the request rate for each virtual network is fixed to 400 requests of movies per hour (400 req/h), and the interval between each request is given by an exponential distribution (this request model is based on [7]). The request rate is kept constant and active during the whole simulation. When a request arrives, the next action is the transmission of the movie. All movies in the experiments have the same size (4 GB).

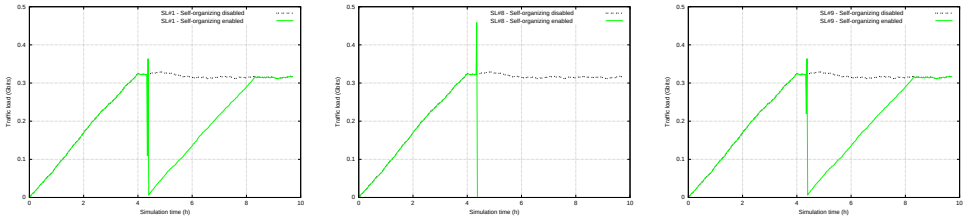
V. Results

The evaluation shows the efficiency of using the self-organizing model in terms of spared network traffic. Using the scenario described above, almost 10 hours of user's request

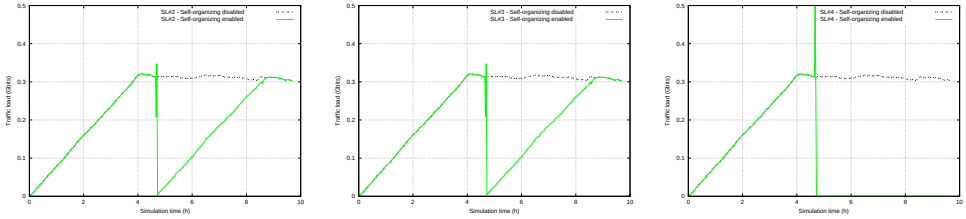


(a) Self-organizing disabled (b) Self-organizing enabled

Fig. 6. Traffic load of all substrate links



(a) VN1 - Substrate Link SL#1 (b) VN1 - Substrate Link SL#8 (c) VN1 - Substrate Link SL#9



(d) VN2 - Substrate Link SL#2 (e) VN2 - Substrate Link SL#3 (f) VN2 - Substrate Link SL#4

Fig. 7. Traffic load of each substrate link used by the testbed

were simulated, being the self-organizing cycle activated every 5 minutes. Traffic load of the substrate links and the average latency of the packets are measured every second stage monitoring interval (1.5 minutes).

A. Traffic Load

The traffic load curves presented in the experiments show an ascending behavior until 4 hours of simulation, and after this the traffic load reaches the maximum average load per virtual link for the request rate of 400 req/h, i.e., around 0.325 Gbits.

During the simulations with the self-organizing model disabled, Fig. 6(a), the substrate links used by VN1 (light-lines) have their curves overlapped because they have the same traffic load. The same overlapping occurs with the traffic load of the substrate links used by VN2 (dark-lines in Fig. 6(a)). The traffic of both virtual networks follows a cut-through pattern and for this reason the traffic load of those substrate links is the same. However, when the self-organizing model is enabled, Fig. 6(b), the traffic load of some substrate links changes. The self-organizing model reallocates the virtual resources in two distinct cycles. The first one happened after 4 hours of simulation and reorganized the virtual resources of VN1. The second cycle happened near 5 hours of simulation, and the virtual resources of VN2 were reorganized.

To understand how the self-organizing model changes the traffic load of the substrate links we present Fig. 7, that shows the traffic load of each substrate link when the self-organizing model is enabled and disabled. The traffic load of each substrate link associated to the VN1 is presented in Figs. 7(a) - 7(c), and the traffic load of the VN2 substrate links is illustrate in Figs. 7(d) - 7(f). Analyzing the traffic load of the

substrate links presented in Fig. 7 it is possible to discover which virtual nodes moved during the self-organizing cycles.

In the case of the VN1, the traffic load of substrate link “SL#8”, Fig. 7(b), indicates that virtual node “VN1#N3” moved from substrate node “H” to “I”, because after 4 hours of simulation the traffic load changes from 0.32 Gbits to 0.46 Gbits and right after this it is interrupted (drops to zero). Associated to this fact, the traffic load of substrate links “SL#1”, Fig. 7(a), and “SL#9”, Fig. 7(c), also increases from 0.32 Gbits to 0.36 Gbits after 4 hours of simulation. This increase occurs because the virtual node “VN1#N3”, now placed at the substrate node “I”, processes all packets queued during the moving phase. In the sequence, the traffic load of “SL#1” and “SL#9” drops to 7.4 Mbits and starts to rise again as soon as the virtual application inside the virtual node “VN1#N3” restarts to transmit movies in response to new requests. Progressively, the traffic load of those substrate links increases until it reaches the maximum average load (0.325 Gbits) after 8 hours of simulation. The same traffic behavior is observed for the VN2. In this case, the virtual node “VN2#N2” moved from the substrate node “E” to “D”, and this caused the elimination of traffic load on the substrate link “SL#4”, Fig. 7(f). The pattern of increasing, dropping, and increasing again is also observed in traffic load of the substrate links “SL#2” and “SL#3”, respectivelly in Fig. 7(d) and Fig. 7(e).

The graphics of Fig. 7 show that using the self-organizing model it is possible to reduce 1/3 of the traffic load when the transmissions on substrate links “SL#8” and “SL#4” are eliminated. Now, to have an entire picture of the amount of network resources spared with the self-organizing model, we present Fig. 8, where the traffic load of all substrate nodes of the scenario is summed and graphically presented. Considering the scenario used on the evaluation, the total traffic load of the network is approximately 1.9 Gbits when the self-organizing model is disabled, and when the model is enabled it reaches the stable state using 1.2 Gbits. This means that 36.8% of the network resources of this scenario were spared when the self-organizing model is applied to managed the network resources.

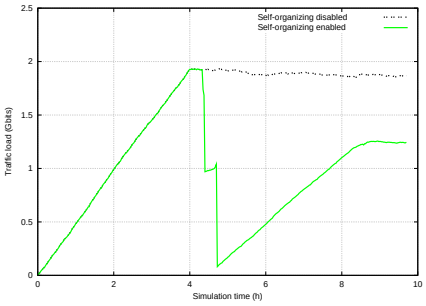
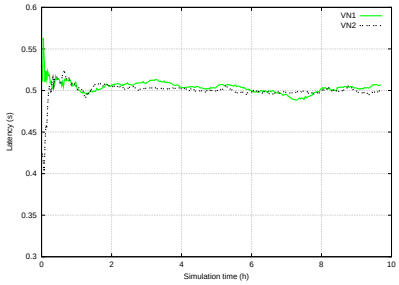


Fig. 8. Sum of traffic load from all substrate nodes

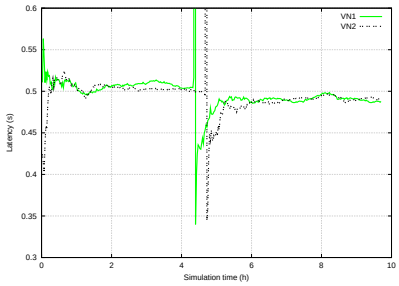
Despite the advantages on using the self-organizing model to managed the traffic load of the substrate network, there are issues regarding at least (i) the interruption time of the applications inside the virtual networks and (ii) the latency of the packets when a virtual node migration is required during the self-organizing cycle. The first issue was addressed in a previous work [6], and it mainly depends on the efficiency of the moving mechanism and the amount of data that has to be moved. The second issue this investigated as follows.

B. Latency

This experiment shows the average latency of packets arrived in the destination virtual node of each virtual network within the monitoring interval (1.5 minutes). Fig. 9 shows the packet latency measured for each virtual network.



(a) Self-organizing disabled



(b) Self-organizing enabled

Fig. 9. Latency of virtual networks flows

The latency of the packets for both virtual network is approximately 0.51 s when the self-organizing model is disabled, as illustrated in Fig. 9(a). The same average latency is observed in Fig. 9(b) until the moment that the first reorganization is executed. It is not illustrated in Fig. 9(b), but the average latency during the period of reorganization associated to the VN1 reaches 50.04 s, and during the reorganization related to VN2 it is approximately 50.84 s. After the reorganization of the virtual resources the average latency remains stable around 0.48 s for both virtual networks. This value represents a reduction of 5.9% of the average latency if compared to the latency before the reorganization.

The self-organizing model can reduce the latency of the packets but the cost associated to this benefit is a high latency (around 50 s) during a period of at least 1.5 min. (as presented in Fig. 9(b)). This high latency might become a problem when the applications running inside the virtual network require strict QoS (Quality of Service) guarantees.

VI. Related Work

Network virtualization is an emerging research area. Most of current researches in this area focus on defining an efficient mapping or embedding process of virtual networks into the substrate network [8][9][10].

Houidi *et al.* [8] presented a distributed and autonomic mapping framework responsible for self-organizing the virtual networks on top of the substrate network every time a new deployment request arrives. Despite the fact this approach employs autonomic features and distribution, the self-organization is subjected solely to the changes on the number of virtual networks running on top of the substrate network. Thus, changes on the amount of resources used by the virtual networks during their lifetime are not explicitly considered.

The work presented by Yuy *et al.* [9] deals with dynamic requests for embedding/removing virtual networks. The authors map the constrains of the virtual network to the substrate network by splitting the requirements of one virtual link in more than one substrate link. A time window is used to regulate when a reorganization of the virtual links is required. The problem of this approach is to define a time window also able to deal with changes on the use of the resource during the lifetime of the virtual networks, and not only with the dynamics on embedding/removing virtual networks.

Chowdhury *et al.* [10] proposed algorithms for embedding a virtual network that correlates both node and link mapping requirements. The process is divided in two phases. First, the virtual nodes are mapped and then takes place mapping process of the virtual links. Once again, this approach deals with the deployment phase of a virtual network, but it is not target to deal with changes on the amount of resources used by virtual networks during their lifetime.

Differently from current researches, the self-organizing model presented in this paper addresses the management of substrate resources during the lifetime of a virtual network rather than the initial embedding phase. The decision of when a self-organization is required is subjected solely to resource consumption conditions of the running substrate network infrastructure. External interferences, like a new virtual network embedding or removing, are not the main issues to be managed, but the effects of any kind of changes on the substrate environment are the kernel of our proposal.

VII. Conclusion

This paper presents a distributed self-organizing model to manage the substrate resources in network virtualization. The main objective of this model is to manage the amount of network resources used during the lifetime of the virtual networks. The triggers of self-organizing actions are the local measurements and neighbor information. The experiments showed that the benefits in terms of reduction of traffic load are more expressive than the results in terms of latency. Moreover, the high latency observed during the reorganization process might reduce the range of types of virtual networks that comply to our self-organizing model.

During the development of this research some issues raised as future investigations. For example, there might exist an economical model to determine the number of virtual pipes and nodes associated to a virtual network. This model actually represents a trade-off between costs on buying slices of resources versus the flexibility on managing the substrate network resources. Other investigations go towards cross-layer interaction to improve the resources management.

Acknowledgement

This work was partly funded by the CNPq Brazilian Research Agency and by the European Union through the 4WARD project in the 7th Framework Programme. The views expressed in this paper are solely those of the authors and do not necessarily represent the views of their employers, the 4WARD project, or the Commission of the European Union.

References

[1] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20 – 26, July 2009.

[2] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley, "Evaluating Xen for Router Virtualization," in *Proceedings of 16th International Conference on Computer Communications and Networks, 2007. ICCCN 2007.*, August 2007, pp. 1256–1261.

[3] Y. Ohsita, T. Miyamura, S. Arakawa, S. Ata, E. Oki, K. Shiomoto, and M. Murata, "Gradually Reconfiguring Virtual Network Topologies Based on Estimated Traffic Matrices," in *Proceedings of 26th IEEE International Conference on Computer Communications. IEEE INFOCOM 2007*, May 2007, pp. 2511–2515.

[4] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: live router migration as a network-management primitive," in *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*. New York, NY, USA: ACM, 2008, pp. 231–242.

[5] N. Degrande, K. Laevens, D. D. Vleeschauwer, and R. Sharpe, "Increasing the User Perceived Quality for IPTV Services," *IEEE Communications Magazine*, vol. 46, no. 2, pp. 94–99, 2008.

[6] C. Marquezan, J. Nobre, L. Granville, G. Nunzi, D. Dudkowski, and M. Brunner, "Distributed reallocation scheme for virtual network resources," in *IEEE International Conference on Communications (ICC 2009)*, 2009.

[7] D. Agrawal, M. S. Beigi, C. Bisdikian, and N. Lee, "Planning and Managing the IPTV Service Deployment," in *Proceeding of 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, 2007, pp. 353 – 362.

[8] I. Houidi, W. Louati, and D. Zeghlache, "A Distributed and Autonomic Virtual Network Mapping Framework," in *Proceedings of Fourth IEEE/IFIP International Conference on Autonomic and Autonomous Systems, 2008. ICAS 2008*, March 2008, pp. 241–247.

[9] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Computer Communications Review*, vol. 38, no. 2, pp. 17–29, 2008.

[10] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proceedings. The 28th Conference on Computer Communications. IEEE INFOCOM 2009*, April 2009, pp. 783 – 791.

Distributed Reallocation Scheme for Virtual Network Resources

Clarissa Cassales Marquezan, Jéferson Campos Nobre, Lisandro Zambenedetti Granville
Institute of Informatics - UFRGS
Porto Alegre, RS, Brazil
Email: {clarissa, jcnobre, granville}@inf.ufrgs.br

Giorgio Nunzi, Dominique Dudkowski, Marcus Brunner
NEC Europe Network Laboratories
Heidelberg, Germany
Email: {nunzi,dudkowski,brunner}@nw.neclab.eu

***Abstract*—Network virtualization is an emerging technology for cost-effective sharing of network resources. The key strategy in network virtualization is of slicing physical resources (links, CPU, memory, and storage) to create virtual networks that are assigned to different operators. One important challenge on network virtualization is the efficient use of the physical resources. To accomplish such efficient use the management of the physical resources should be transparent to the applications running within the virtual networks, and should be executed at runtime in order to deal with the variation on the load requests of different virtual networks. Traditional resource allocation schemes use offline, centralized, and global view strategies to manage the use of physical resources. In contrast to these strategies, we propose a runtime, distributed, local view approach to manage physical resources. In this paper we introduce a virtual network architecture and an associated self-organizing algorithm to reallocate virtual network resources along different physical nodes in order to equalize the bandwidth, and storage consumption on the physical nodes. We developed a virtual network model based on Omnet++ to simulate the designed self-organizing algorithm. An IPTV testbed scenario is presented and initial experiments, about the interruption time of the application inside the IPTV virtual network, are described.**

I. Introduction

The increasing demand of multimedia services over the Internet is pushing for new methods to allocate resources in future networks. For example, IPTV services are expected to become more and more popular and integrated offers, like the triple-pay packages, require cost-effective strategies for resource allocation. In fact, a typical IPTV network infrastructure requires significant investments for the distribution network, in terms of guaranteed bandwidth as well as available storage capacity. Normally, these resources need to be planned and well dimensioned in advance, before upper services can be actually deployed [1].

The costs of deploying a physical infrastructure may prevent many service providers to get into the market, like in the case of IPTV services [2]. Nevertheless, recent works in the field of virtual networks offer a viable alternative that promises to cut costs by sharing the infrastructure among different service providers [3]. The key on network virtualization is of dividing the physical network infrastructure into several slices and associating them to different virtual providers. The deployment of virtual networks must observe two different perspectives. The former is the perspective of a virtual provider, who wants the accomplishment of the contracted resources (SLAs must be maintained), while the later regards to the physical infrastructure provider, who wants to save as much as possible its physical resources in order to maximize revenues.

Being this, efficient algorithms to allocate physical resources (links, CPU and storage capacity) must be put in place by physical providers, otherwise punctual high loads on multiplexed physical resources may create resource scarcity that can prevent the deployment of new virtual networks. Traditionally, physical resources are allocated in the initial planning phase: a planning tool [4] provides the estimated dimensioning of network components given a certain SLA and resources are allocated based on this output. This approach can be applied for small virtual environments, but in large scale deployments a static allocation cannot take in account the mass imbalance of users requests between different locations.

In order to efficiently consume resources of the physical infrastructure, this paper proposes a real-time reallocation of virtual network resources. The main contribution of this paper is twofold. First, we propose a new approach to the deployment of services of virtual networks: with this approach, resources can be dynamically moved within the virtual layer to maximize over time the consumption of physical resources. Second, we define a distributed algorithm based on self-organizing techniques to implement a real-time reallocation scheme for virtual networks.

The proposed virtual model was implemented in the Omnet++ simulator and we defined an IPTV scenario with virtual providers in order to test the self-organizing reallocation scheme. The objective of the simulation, in this paper, is identifying the impact of the moving process in terms of interruption of IPTV services.

The reminder of this paper is organized as follows. Section 2 brings the related work on self-organization and virtual networks. Section 3 presents the proposed virtual network model and section 4 presents the designed reallocating scheme. Section 5 presents the implementation of the proposed model using the Omnet ++ simulator, while section 6 describes the testbed scenario. The evaluation and the associated results are discussed in Section 7. Finally, the conclusions and future work are presented in Section 8.

II. Related Work

On a first sight, the proposed self-organization reallocation scheme might be seen as an extension of existing virtual machine live migration. Recently, self-organization techniques have been employed on server virtualization scenarios [5][6]. In these cases, the virtual machines are self-organized according to the workloads of the physical nodes, and generally, this self-organization is accomplished migrating virtual machines to physical ones with lower workloads. However, the metrics traditionally used to determine the workload of virtual machines are CPU and memory, and in a virtual network the bandwidth consumption is one major metric to be considered in the migration process. Beyond these metrics, virtual network live migration is different from virtual machine migration because it has also to deal with virtual topology issues and routing connections reconfigurations.

The research presented by Yuichi Ohsita et al. [7] is able to make the reconfiguration of the virtual network topology in order to cope with the traffic demands. The authors use traffic matrix estimation, and partial view of the virtual nodes to make the reconfiguration decisions. A sub set of the authors from this first paper, Takashi Miyamura et al. [8], enhanced the previous research and defined a centralized server devoted to identify traffic on demand fluctuation and network failures. Based on this, a virtual network reconfiguration is activated. Both cases, the re-configuration is just restrict to links of a virtual network and does not consider that this process might involve migration of an entire virtual device, like a router.

A recent research on virtual router migration is presented by Yi Wang et al. [9]. In this paper the authors proposed a virtual router migration mechanism, where virtual interfaces of the routers are not directly mapped to physical ports and in this sense it is possible to migrate a router among different physical devices. The authors presented the migration mechanism itself and the advantages of using this approach to deal with management changes, planning, and new service deployment. However, nothing was mentioned about the analysis to trigger the router migration, and how this approach can help to reduce punctual high loads on the physical infrastructure.

Based on the aforementioned we believe that current researches do not address the problem of reallocate virtual resources at runtime, using local view, and based on a distributed approach. The next sections present the proposed solution.

III. Virtual Network Model

According to recent researches, virtualization is a promising technique to deploy future networks [3][10][11]. Its key idea is the identification and separation of two roles: a physical provider, who owns and maintains the physical network, and a virtual provider, who builds its own infrastructure by renting slices of resources from the physical provider. If we look at a virtual provider as an entity selling services, the advantage of virtualization relies on the fact that costs in running a physical infrastructure can be outsourced to an external provider.

For this paper, it is important to describe the main characteristics of an architecture for virtualization; it should be noted that this paper presents the minimal assumptions, and further details can be found in specific projects like GENI or 4WARD [12]. The physical resources of a node are sliced into different virtual nodes: each virtual node is assigned to a different customer. Physical resources include CPU power, memory space, storage capacity, network interfaces and bandwidth.

An important aspect in the architecture of virtual networks is the transparency: virtual nodes cannot see or exchange any type of information, in order to assure isolation of the networks of different providers. Additionally, the data exchanged in the virtual network is transparent to the physical provider to preserve the privacy of the customers. Nevertheless, some minimal primitives to inspect the activity of the different slices are normally available: as an example, primitives to allow the controller of the physical resources to know the actual usage of computational resources and traffic consumption. Figure 1 shows the architectural view of a node, where resources are sliced and assigned to different virtual providers.

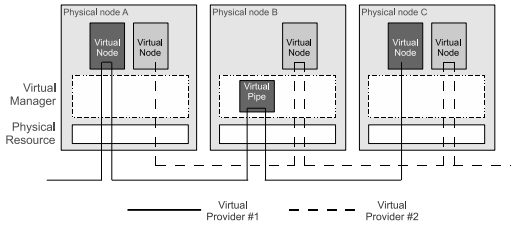


Fig. 1. Virtual node

A physical node is composed of: physical resources, virtual manager, virtual nodes, and virtual pipes. The virtual manager is responsible for receiving the requests for deployment of a virtual node or pipe and managing locally the connections and resources associated to virtual network. The virtual manager can be seen as the “hypervisor” concept used on virtual machines technologies, for instance.

A virtual node is a slice of the physical node comprehending: CPU power, memory space, application(s), storage capacity (if necessary), network interface and bandwidth multiplexing. A virtual pipe regards all virtual node features but application and storage capacity. The introduction of the virtual pipe concept supports the creation of virtual links between non-adjacent virtual nodes. Figure 2 illustrate the differences of using virtual nodes and virtual pipes considering the physical view and the virtual views.

The technology for creating virtual links is already available on current routers [9], but we believe that it is necessary to have some mechanism to determine the amount of traffic passing by physical connections that compose the virtual link, in order to enable a better management of the virtual networks resources. For example, the employment of virtual pipes allows the virtual manager to identify forward traffic inside a physical node without inspecting the packets belonging to a virtual network associated to this traffic. In our model, the

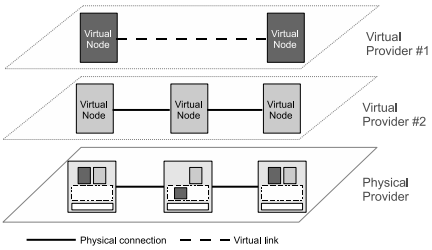


Fig. 2. Virtual link representation

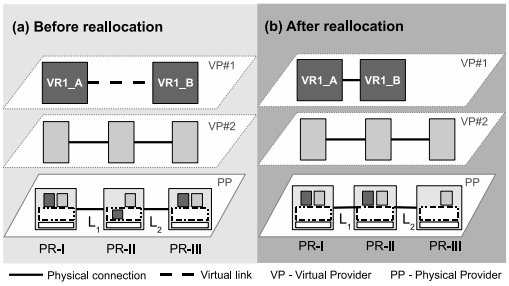


Fig. 3. Reallocation scheme

information associated to this kind of traffic is one of the inputs used to analyze the necessity of reallocating virtual resources. In the next section we present our solution for efficient resources reallocation.

IV. Distributed Reallocation Scheme

As mentioned before the major contributions of our proposal are the employment of distribution, local view, and online features on the reallocation of resources of virtual networks. Some assumptions must be observed in order to provide such features in the new scheme, and they are described below.

- The initial deployment of a virtual network is not addressed by this work, and we assume that a different, external planning tool analyzes the conditions of physical resources and then choose the best initial placement for the slices of the new virtual network.
- We assume that the virtual topology defined by the first placement will not change during the lifetime of the virtual network, even after the reallocation of virtual slices among physical nodes.
- The reallocation of slices must be as transparent as possible for the virtual node. In the current stage of this research, the reallocation of the virtual slices is transparent in the sense of avoiding to exchange any kind of information between the virtual application inside the moving slice and the virtual managers of the physical nodes involved in the reallocation operation. However, we introduce an interruption time on the execution of the application running inside the moving virtual slice.

Based on these assumptions, and inspired on self-organization techniques presented in [13] we defined a reallocation scheme that is executed locally by each virtual manager inside the physical nodes. The main objective of this mechanism is to approximate the virtual node that is generating a great amount of traffic to the destination virtual node. The approximation is done moving the source virtual node from its physical device to another physical device near the destination virtual node. Figure 3 illustrates the reallocation of a virtual router of an IPTV virtual provider (details about the IPTV infrastructure can be found in [2]).

The algorithm used to accomplish the reallocation scheme is divided in five stages. First, locally each virtual manager analyzes the existence of some traffic associated to a virtual node with characteristics to be moved. We defined heuristics to identify traffic patterns overloading links of the substrate network. These heuristics correlate information of local resources, such as incomming/outgoing traffic and the amount of memory/storage read and write to determine if there is some cut-through traffic that should be eliminated by moving/receiving a virtual node[1]. On the second stage, the physical neighbors exchange information about the virtual nodes that must be received or moved. Locally, on the third stage, each neighbor analyzes the exchanged information, and the physical node that must move a virtual resource decides to whom the virtual resources might be moved. The forth stage is the decision and the reservation of the resources at the target physical neighbor. Finally, the virtual resources are moved.

During the third and fifth stages the application(s) running inside the virtual node are suspended, and all packets related to this virtual node are queued by the virtual manager. As soon as the virtual node is reestablished on the physical neighbor the packets are unqueued and sent to the virtual node on the new physical location. As aforementioned, the proposed reallocation scheme imposes an interruption time on the application running on the virtual node. This interruption time depends on the nature of virtual node that is being moved. For example, if the virtual node is an IPTV router, the interruption time might be higher because the storage associated to the IPTV router must be also moved. On the other hand, if the virtual node is a common router the interruption time should not be prohibitive because less resources should be moved. Discussions about the routing process during the migration of virtual routers are out of the scope of this paper.

According to the description provided above, it is possible to observe that our proposal does not need a global view of the physical topology to identify the overloaded physical resources, like links or devices. Just using the local information retrieved from the controllers of network interfaces, CPU, memory, and disk, our heuristic is able to identify possible virtual candidates to be moved. Moreover, we also do not

[1]Due to space limitations the heuristics developed to identify the virtual nodes to be moved will not be presented in this paper.

need any centralized entity to make the decision of reallocating resources. Our approach is completely distributed and based on information exchanged among the physical neighbors the reallocating scheme is triggered. On the next sections we present the implementation, testbed, and evaluation of the proposed reallocation scheme.

V. Implementation

To validate our reallocation scheme we implemented a new module for Omnet++ Simulator. This new module is presented in Figure 4. The network presented in this figure is composed of 5 physical nodes and 2 virtual IPTV providers ("vnetA" and "vnetB"). Most of the parameters of this virtual module are configurable, like for example the number of physical devices, virtual nodes, and pipes, and also the features of these elements. However, the current version of this module does not support the definition of different network topologies, and only ring network topology can be described in this version.

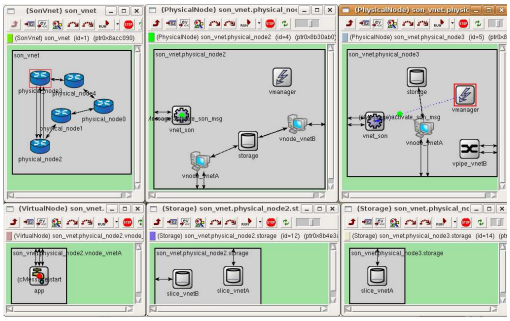


Fig. 4. Virtual module for Omnet++ simulator

So far, in this paper we show the execution of the proposed distributed virtual reallocation scheme in the light of the reallocation of virtual resources from an IPTV virtual provider. To accomplish this we defined a virtual network where an IPTV provider deploys the required infrastructure to attend the requests for movie streams of their costumers, and the associated testbed is described in the sequence.

VI. Testbed

We consider a scenario where the IPTV provider requires routers connecting costumers, and the planning tool, responsible for defining the first placement of the virtual resources, has allocated two virtual routers in different physical routers. Furthermore, storage slices have been attached to each virtual router. These two virtual nodes are connected through a virtual link and belong to the virtual provider #1 (VP#1) illustrated in Figure 3.

In the evaluated scenario, users from both virtual providers (VP#1 and VP#2) are requesting movies and thus generating traffic in the physical links L_1 and L_2. The experiments consider that the traffic load imposed by the requests from the users connected to the virtual router "VR1_A" to the virtual router "VR1_B" is higher then any other traffic on the network presented in Figure 3. So far, after the local execution of the reallocation schema by the virtual managers of each physical device, the virtual node "VR1_B" (at physical device "PR-III") is identified as the candidate to be moved to the physical device "PR-II".

To run the simulation some main parameters are required and Table I presents these parameters and the associated values used on the simulation.

Parameter	*Associated value*
Datarate of links associated to each virtual network	1 Gbps
Delay of of links associated to each virtual network	1 ms
Datarate of storage	100 Mbps
Delay of storage	1 ms

TABLE I
Parameters of the simulation

Being this, the initial experiments proposed in this paper investigate the interruption time to move the virtual router "VR1_B" and the storage directly connected to it, when the size of the storage varies from 1GB to 10GB. We also discuss the compromise between maintaining a low interruption time and the number of virtual resources necessary to keep the fixed interruption time.

VII. Evaluation

Figure 5 presents the graphic of the interruption time associated to the scenario described above. The interruption time is composed of time to: (a) exchange the control messages between the physical neighbors in order to reserve the required resources for the reallocation process; (b) read and send data from one storage to the storage of the physical neighbor; and (c) write data on the storage of the new physical location of virtual router "VR1_B". The interruption time increases linearly with the increase of the storage size, as expected.

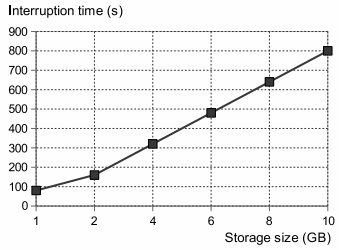


Fig. 5. Interruption time

The analysis of the interruption time is not so interesting when it is done in an isolated fashion. However, the analysis of the interruption time in the light of the amount of virtual

resources used by a virtual provider can become a business metric at the moment that a virtual provider is contracting a virtual network. For example, let's consider the scenario where an IPTV provider requires a storage capacity of 20 GB, and the features described in Table I are being used. In this case, the smallest interruption time (i.e, 80s considering that the minimum storage slice is 1GB) is guaranteed when 20 virtual nodes are used.

In this sense, the maintenance of low interruption time during the reallocation scheme imposes the deployment of more virtual resources to a single virtual provider. This information can be used by both sides, physical and virtual providers, to determine the behavior of the reallocation process. For instance, if the virtual provider contracting a virtual network does not desire high interruption time on the applications running inside the virtual network, it can force the physical provider not to employ the reallocation scheme to this virtual network. However, the physical provider can increase the prices for virtual providers that want more fixed constrains on the maintenance of the virtual network operation. We believe that this tradeoff is a metric to be agreed on the SLA between the physical and the virtual providers before the deployment of the virtual network.

VIII. Conclusion

This paper presented the definition of a distributed reallocation scheme for virtual network resources, a high level architecture for virtual networks, and first experiments using the reallocation scheme on IPTV scenario. The main objectives of the experiments were presenting the correct execution of the reallocation mechanism, identifying the interruption time of the applications (inside the virtual nodes) imposed by the resource reallocation process, and analyzing the relationship between interruption time and virtual resources composing the virtual network.

The major outcome of this initial experiment is the utilization of the interruption time and number of virtual resources in order to determine the terms of the SLA between the physical and virtual providers. If application outages are not a constrain for the virtual provider, it is possible to firm an SLA giving more flexibility to the physical provider reallocates the virtual resources, and this flexibility might be translated into a reduction on the price of the virtual network deployment. The major benefit in this case stays with the physical provider, that can reallocate the virtual resources in order to efficiently use the physical resources. On the other hand, virtual provides that require restrict reallocation policies and low interruption time, would not allow the employment of the reallocation scheme, as a consequence the costs for the virtual provider might be increased.

As future work we intend to extend the experiments using the IPTV virtual networks, and employ a user request model to verify the full operation of the reallocation scheme. The next evaluation scenario aims to identify the costs of moving virtual resources, considering the relationship between interruption time and the saved bandwidth on the physical links after the execution of the reallocation scheme. We also intend to test the behavior of other kind of applications on top of the virtual model, for instance, network management applications. Furthermore, we intend to investigate how self-* features, like self-healing, self-configuration, self-awareness, self-monitoring, can improve the management of the virtual networks on top of the physical network.

Acknowledgement

This work was partly funded by the Brazilian Ministry of Education (MEC/CAPES, PDEE Program, process 4436075), and by the European Union through the 4WARD project in the 7th Framework Programme. The views expressed in this paper are solely those of the authors and do not necessarily represent the views of their employers, the 4WARD project, or the Commission of the European Union.

References

[1] N. Degrande, K. Laevens, D. D. Vleeschauwer, and R. Sharpe, "Increasing the User Perceived Quality for IPTV Services," *IEEE Communications Magazine*, vol. 46, no. 2, pp. 94–99, 2008.

[2] S. Han, S. Lisle, and G. Nehib, "IPTV Transport Architecture Alternatives and Economic Considerations," *IEEE Communications Magazine*, vol. 46, no. 2, pp. 70–77, 2008.

[3] N. Niebert, I. E. Khayat, S. Baucke, R. Keller, R. Rembarz, and J. Sachs, "Network Virtualization: A Viable Path Towards the Future Internet," *Journal Wireless Personal Communications*, vol. 45, no. 4, pp. 511–520, June 2008.

[4] D. Agrawal, M. S. Beigi, C. Bisdikian, and N. Lee, "Planning and Managing the IPTV Service Deployment," in *Proceeding of 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, 2007, pp. 353 – 362.

[5] M. Steinder, I. Whalley, D. Carrera, I. Gaweda, and D. Chess, "Server virtualization in autonomic management of heterogeneous workloads," in *Proceeding of 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, May 2007, pp. 139–148.

[6] X. Y. Wang, D. Lan, X. Fang, M. Ye, and Y. Chen, "A resource management framework for multi-tier service delivery in autonomic virtualized environments," in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2008)*, 2008, CDROM.

[7] Y. Ohsita, T. Miyamura, S. Arakawa, S. Ata, E. Oki, K. Shiomoto, and M. Murata, "Gradually Reconfiguring Virtual Network Topologies Based on Estimated Traffic Matrices," in *Proceedings of 26th IEEE International Conference on Computer Communications. IEEE INFOCOM 2007*, May 2007, pp. 2511 – 2515.

[8] T. Miyamura, E. Oki, I. Inoue, and K. Shiomoto, "Enhancing bandwidth on demand service based on virtual network topology control," in *IEEE Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008.*, April 2008, pp. 201–206.

[9] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: Live router migration as a network-management primitive," in *ACM SIGCOMM Computer Communications Review*, August 2008, pp. 231–242.

[10] N. Feamster, L. Gao, and J. Rexford, "How to lease the internet in your spare time," in *ACM SIGCOMM Computer Communications Review*, Jan 2007.

[11] A. Berl, A. Fischer, H. de Meer, A. Galis, and J. Rubio-Loyola, "Management of Virtual Networks," in *Proceedings of Fourth IEEE/IFIP International Workshop on End-to-End Virtualization and Grid Management, EVGM 2008*, September 2008, pp. 197 – 202.

[12] 4WARD, "The fp7 4ward project," 2008, available at http://www.4ward-project.eu/.

[13] K. Hermann, "Self-organizing replica placement - a case study on emergence," in *IEEE First International Conference on Self-Adaptative and Self-Organizing Systems (SASO 2007)*, 2007.

Maintenance of Monitoring Systems Throughout Self-Healing Mechanisms

Clarissa Cassales Marquezan[1,2], André Panisson[1], Lisandro Zambenedetti Granville[1], Giorgio Nunzi[2], Marcus Brunner[2]

[1] Federal University of Rio Grande do Sul - Porto Alegre, Brazil
{clarissa, panisson, granville}@inf.ufrgs.br
[2] NEC Europe Network Laboratories - Heidelberg, Germany
{marquezan, nunzi, brunner}@nw.neclab.eu

Abstract. Monitoring is essential in modern network management. However, current monitoring systems are unable to recover their internal faulty entities forcing the network administrator to manually fix the occasionally broken monitoring solution. In this paper we address this issue by introducing a self-healing monitoring solution. This solution is described considering a scenario of a monitoring system for a Network Access Control (NAC) installation. The proposed solution combines the availability provided by P2P-based overlays with self-healing abilities. This paper also describes a set of experimental evaluations whose results present the tradeoff between the time required to recover the monitoring infrastructure when failures occur, and the associated bandwidth consumed in this process. Based on the experiments we show that it is possible to improve availability and robustness with minimum human intervention.

1 Introduction

Network and service monitoring is an activity essential to identif problems in underlying IT communication infrastructures of modern organizations. Monitoring is typically materialized by systems that periodically contact elements (*e.g.*, network devices and services) to check their availability and internal status. A monitoring system may be simple (like the Multi Router Traffic Grapher (MRTG) [1]) or complex, being composed of as diverse entities as monitors, agents, and event notifiers. The information collected and processed by monitoring systems enables human administrators, responsible for managing the IT infrastructure, to identify (and possibly predict) problems, and thus react in order to keep the managed infrastructure operating in a proper way.

Monitoring systems must run uninterruptedly to ensure that failures in the managed elements are detected. Problems in the monitoring systems break the monitoring process and can lead the human administrator to believe that the managed elements are working properly even when they are not. Robust monitoring systems should thus employ mechanisms not only to identify failures on the managed infrastructure, but also to recover the faulty monitoring solution itself. Currently, however, most monitoring systems force the administrator to manually recover the occasionally broken solution. Such a manual approach may not drastically affect the monitoring of small networks, but in larger infrastructures the approach will not scale and should be replaced efficient

alternatives. The self-managed approach is one alternative emerging as a solution for the manual approach. Typically, a self-managed system is built on top of self-* features capable to reduce the human intervention and provide more efficient results.

In this paper we address the problem of monitoring systems that lack self-healingness feature by considering the example of a Network Access Control (NAC) [2] installation. A NAC secured network is composed of devices and services (*e.g.*, routers, firewalls, RADIUS servers) that control how users and devices join the network. Traditional monitoring systems (*i.e.*, without self-healing support) fail to protect NAC, for example, in two situations. First, consider a crashed RADIUS server whose associated RADIUS monitor crashed too. In this case, the administrator reacts to the RADIUS problem only when users complain about unsuccessful login attempts. Worse than that, however, is the second situation. Suppose a failure in the *rogue user* service responsible for detecting unregistered devices, and another failure in the monitor associated to it. In this case, unregistered devices will silently join the network without generating user complains. In contrast to the first situation, the "silent failure" remains because no signal is issued either by network users or, and most seriously, by the now broken monitoring system.

Recent researches on autonomic management certainly present self-* concepts that could be used to address the aforementioned problems. Such researches, however, take a mostly abstract approach and rarely touch concrete implementation issues. In this paper, in turn, we investigate the employment of self-* features to actually implement, deploy, and evaluate a self-healing monitoring system able to recover from internal failures without requiring, at some extend, human intervention. The goal of our research is to understand the advantages and drawbacks of using self-* features in a real scenario of a service monitoring system.

The monitoring elements of our solution implement two main processes: regular monitoring (to monitor final devices and services) and recovery (to heal the monitoring system). We evaluate our solution in terms of recovery time when fail-stop crashes occur in the monitoring system. In addition, we also measure the traffic generated by the communication between the elements of our solution. This leads us to the contribution of showing the tradeoff between the recovery time and associated network traffic. The determination of such tradeoff is important because it shows when a faster recovery process consumes too much network bandwidth. On the other side, it also shows when excessively saved bandwidth leads to services that remain unavailable longer.

The remainder of the paper is organized as follows. In Section 2 we review network monitoring systems in terms of self-* support, distribution, and availability. In Section 3 we introduce our self-healing architecture for NAC monitoring, while in section 4 we evaluate our proposal in an actual testing environment, presenting associated results and their analyses. Finally, the paper is concluded in Section 5.

2 Related Work

Although network and service monitoring is widely addressed by current investigations [3] [4] and products on the market [5], we review in this section solutions that are mainly related to the aspects of self-monitoring and self-healing on distributed monitoring.

Distributed monitoring are specially required in large-scale networks, like country or continental-wide backbones [6] [7]. Most of the research in this area propose complex monitoring system that generally identify internal failures and employ algorithms to reorganize itself without the failed components. In this sense, the solutions present a certain level of self-awareness and adaption, although self-healing is in fact not present, *i.e.,* failing entities are not recovered or replaced. It means that in scenarios where most of the monitoring entities crash, the monitoring systems stop working because no mechanism is employed to maintain the execution of the monitoring entities.

Yalagandula *et al.* [8] propose an architecture for monitoring large networks based on sensors, sensing information backplane, and scalable inference engine. The communication among the entities relies on a P2P management overlay using Distributed Hash Tables (DHTs). Prieto and Stadler [9] introduce a monitoring protocol that uses spanning trees to rebuild the P2P overlay used for the communications among the nodes of the monitoring system. Both Yalagandula *et al.* and Prieto and Stadler work can reorganize the monitoring infrastructure if failures are detected in monitoring nodes. After such reorganization the failing nodes are excluded from the core of the rebuilt monitoring infrastructure. Although reorganized, with a few number of nodes, the monitoring capacity of the system is reduced as a whole. Again, adaptation in the form of infrastructure reorganization is present, but proper self-healing it is not.

Few investigations in fact explicitly employ autonomic computing concepts in system monitoring. Chaparadza *et al.* [10], for example, combine self-* aspects and monitoring techniques to build a traffic self-monitoring system. The authors define that self-monitoring networks are those that autonomously decide which information should be monitored, as well as the moment and local where the monitoring task should take place. Nevertheless, such work does not define how the monitoring system should react in case of failures in its components. The meaning of self-monitoring in this case is different than the one of our work. While self-monitoring in Chaparadza's work means autonomous decision about the monitoring process, in our view self-monitoring is about detecting problems, through monitoring techniques, in the monitoring system itself.

Yangfan Zhou and Michael Lyu [11] present a sensor network monitoring system closer to our view of self-monitoring. The authors' contribution resides on the use of sensors themselves to monitor one another in addition to performing their original task of sensing their surrounding environment. Although self-monitoring is achieved, given the restrictions of the sensor nodes (*e.g.* limited lifetime due to low-capacity batteries) the system cannot heal itself by reactivating dead nodes.

Considering the current state of the art, there is a necessity for new approaches for self-monitoring systems. New proposals should explicitly include, in addition to self-awareness already available in the current investigations, self-healing support on the monitoring entities in order to autonomously keep the monitoring service up. In the next section we thus present our self-healing approach for monitoring infrastructures.

3 Self-Healing Architecture for Monitoring Infrastructures

The self-healing architecture built in our investigation forms a P2P management overlay on top of the monitored devices and services. The usage of P2P functionalities in

our architecture provides a transparent mechanism to enable communications target to publish, discover, and access management tasks inside the overlay. In this way, the control of such basic communications is delegated to the P2P framework used to implement our architecture. Furthermore, in a previous work we have presented that network management based on P2P can aggregate benefits, like reliability and scalability, on the execution of management tasks [12]. Now, we use P2P overlays to have the transparency on basic overlay operations, to distribute the identification of failures and also to provide scalability on the recovery process.

The overlay proposed in previous work is called ManP2P and its architecture has been described in the work of Panisson *et al.* [13]. In this current paper, we extend the ManP2P functionalities in order to explicitly support self-healing processes. We believe that combined, self-healing and P2P overlays can bring together a self-monitoring infrastructure able to address current problems on monitoring systems. In this section we review the ManP2P architecture, present self-healing extensions, and exemplifies their employment in the concrete scenario of a NAC installation.

3.1 P2P Management Overlay and Services

The collection of management peers forms the ManP2P management overlay. Each peer runs basic functions (*e.g.*, granting access to other peers to join the overlay or detecting peers that left the P2P network) to maintain the overlay structure. In addition, each peer hosts a set o *management services* instances that execute management task over the managed network. In our specific case, such tasks are monitoring remote equipments. A management service is available if at least one single instance of it is running on the overlay. More instances of the same service, however, must be instantiated in order to implement fault tolerance. Figure 1 exemplifies a scenario where management services (LDAP monitors, Web servers monitors, rogue user monitors) for a NAC installation are deployed on the ManP2P management overlay.

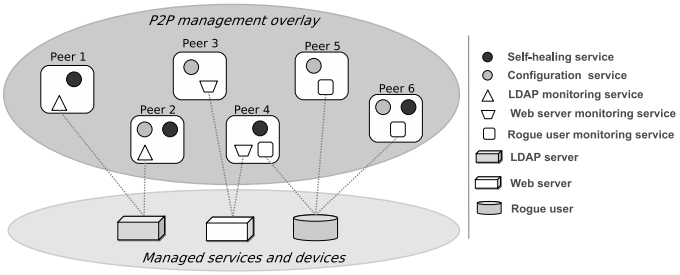


Fig. 1. NAC meta-monitoring infrastructure

In Figure 1, peers #1 and #2 host service instances, pictured as a triangle, that monitor an LDAP server. Peer #4, on its turn, contacts both the Web server and the rogue user service because it hosts management services to monitor these elements. The special services of self-healing and configuration, depicted as black and gray circles respectively, will be explained farther in this paper. Each peer, in summary, may host different services at one. In the extreme cases, there could exist peers with no management services (thus useless peers) or peers hosting one instance of each available management service (this possibly becoming an overloaded peer).

We consider that the a management service is able to heal itself if, after the crashing of some of its instances (possibly due to peers crash), new instances become available, thus recovering the service and guaranteeing its availability. In order to cope with that, two functions must be supported: failure detection and service instance activation.

3.2 Failure Detection

Failures in a management service are detected by a self-monitoring procedure where each service instance, in intervals of t seconds, sends a signal (heartbeat) to all other instances of the same service to inform that the former is running. Self-monitoring, in this sense, means that there is no external entity monitoring the instances of a management service deployed inside the overlay. Indeed, the instances of the management service themselves can monitor their liveness throughout the heartbeat messages. So, if one instance crashes, the other instances will miss the former's heartbeats and then will initiate the process to recover this instance, as it will be explained later on this paper.

Heartbeats that get lost in the network may wrongly suggest the unavailability of a service instance. Instead of immediately assuming an instance as down given the lack of a heartbeat, it first becomes suspect by the other instances. In order to double check the availability of the suspicious instance, one of the other alive instance tries to contact the suspicious instance back. If no contact is possible, the suspicious instance is finally declared unavailable. Assuming s as the time spent to double check the availability of a suspicious instance, the maximum detection time is $td = t + s$.

The distribution of heartbeats from one service instance to all others is accomplished using group communications. At the network level, in the best case, group communication is supported by multicast communications. In this case, the number of heartbeat messages h issued by i service instances in t seconds will be $h = i$. However, if multicasting is not available, the notifying service instance is forced to send, via unicast, copies of the same heartbeat to all other instances. In this case, the number of messages will be $h = i^2 - i$. In this way, the presence of multicasting directly influences the network traffic generated by the failure detection function.

Failure detection is essentially a consensus problem. Solutions on this topic, coming from the dependability field, could be employed and formalisms used to model and validade our detection approach. Instead of that, however, we preferred to use the practical approach of actually implementing the aforementioned function. Although no formal proof is provided, our experiments have shown that this approach is effective in detecting failures in the management service instances.

3.3 Service Instance Activation and Policies

Instance activation is crucial to recover the management service that just lost some of its instances. It is on instance activation that the self-healing and configuration services, presented in Figure 1, play a key role.

Once an instance detects a remote crashed one, it notifies the *self-healing service* that determines how many, if any, new instances of the faulty service must be activated. To do so, the self-healing service internally checks a repository of service policies that describes, for each management service, the minimum number of instances that must be running, as well as the number of new instances that must be activated once the minimum boundary is crossed.

Table 1. Service policy repository

Management service	*Minimum instances*	*Activate instances*
LDAP monitor	2	1
Web server monitor	2	2
Rogue user monitor	2	1

Table 1 shows the service policy repository for the NAC installation of Figure 1. As can be observed, the LDAP monitoring service must have at least 2 instances running. In cause of failure, another new one instance must be activated. In the case of the Web server monitor, on the other hand, although 2 instances are running, whenever activation is required 2 other new instances will be initiated. If the number of remaining running instances of a services is still above the minimum boundary, the self-healing service ignores the faulty service notifications. For example, in the case of the rogue user monitor from Figure 1, if a single instance crashes no action will be executed because the remaining 2 instances do not cross the minimum boundary. Although it is outside the scope of this paper stressing the administration and usage of management service policies (refer to the work of Marquezan *et al.* [14] for that), we assume that policies are defined by the system administrator and transferred to the self-healing service instances long before any failure occurred in the P2P management overlay.

Once required, the self-healing service tries to activate the number of new instances defined in the service policy by contacting the *configuration service*. Such configuration service is then responsible for creating new instances of the faulty service on peers that do not have those instances yet. A peer hosting solely a configuration service can be seen as an spare peer ready to active new instances of any service in failure.

Different than the failure detection function, instance activation is performed outside the group of instances that implement the failing management service. That is so because decoupling the instance activation function from the services that require them allow us to more flexibly deal with the number of components for each function. That directly impact on the number of message exchanged in the overlay.

So far, we have defined a self-healing architecture that extends the ManP2P functionalities. However, to ensure that the failure detection and instance activation functions work properly, two requirements must be filled on the P2P management overlay.

First, each management service (including the self-healing and configuration services) must run at least 2 instances in order to detect and recover problems on the management service. That is so because a single faulty instance cannot react itself if it is crashed, then at least another instance is required. Second, each peer must not host more than one instance of the same management service in order to avoid several instances of that service crashing if the hosting peer crashes too. We assure that the maintenance of the monitoring infrastructure can be accomplished while these requirements are fulfilled.

3.4 System Implementation

As mentioned before, our architecture extends the ManP2P system. The implementation of our architecture in an actual monitoring system is than based on the previous code of ManP2P. Figure 2 left depicts the internal componentes of a peer of our solution.

Components are divided by the *core peer plane* and *management service plane*. The core peer plane's components are responsible for controlling the communication mechanisms between peers. At the bottom the *JXTA* and *network multicast* components implement group communication using unicast (via JXTA) or network multicast. On top of them, the *group manager* and *token manager* components control, respectively, group membership and load balancing (via a virtual token ring). Messages are handled by the *message handler* component that interfaces with Axis2 to communicate with the management service plane's components. A ManP2P component on the top of the core peer plane is used to implement complementary functionalities that are not inside the scope of this paper.

At the management service plane the regular monitoring services are found. Although located in this plane, monitoring services themselves do not monitor remote instances for fault detection; this verification is in fact performed by the group manager component. That is so because we wanted the self-monitoring function to be native in any peer, freeing the developer of new management services to concentrate their efforts on the management functionalities he/she is coding without worrying about self-monitoring. At the management service plane the self-healing and configuration services are also found. As mentioned before, they are responsible for activating new instances of monitoring services when required. The black little square inside the self-healing service represents the policies that define the minimum number of instances of each management service, as well as the number of new instances that must be activated. Peers and internal monitoring services have been coded in Java using Axis2, JXTA, and ManP2P previously developed libraries. Monitoring services have been specifically developed as dynamic libraries that can be instantiated when required be a hosting peer.

4 Experimental Evaluation

In our experimental evaluation we measured the recovery time and the generated network traffic when fail-stop crashes occur in peers of the proposed self-healing monitoring infrastructure. We evaluate the effects of such failures considering variations on: (a) the number of simultaneously crashing peers, (b) the number of peers in the management overlay, and (c) the number of management services running on the overlay.

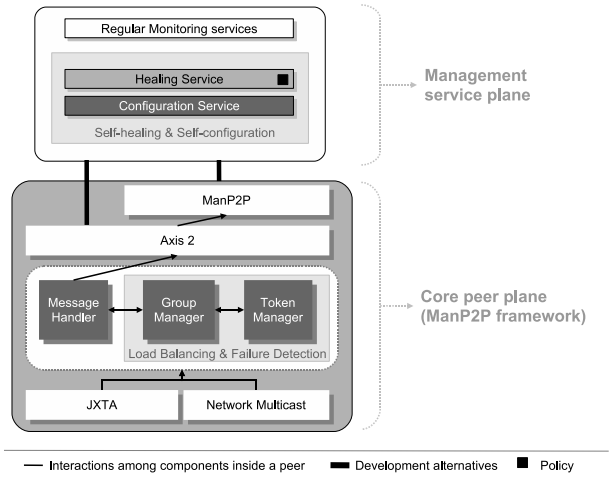


Fig. 2. Meta-monitoring architecture

We have run our experiments in a high performance cluster, called LabTec from the GPPD research group at UFRGS [15], from which we used 16 nodes to host the management peers of our architecture. The recovery time and the generated traffic have been measured capturing the P2P traffic and timestamping it using a packet capture `tcpdump` software. Traffic volume is calculated considering the headers and payload of all packets generated by the system operations. Recovery time has been measured 30 times for each experimental case and computed with a confidence interval of 95%.

Although the size of P2P systems is typically of scales much higher than 16 nodes, we emphasize here that we do not believe that, in an actual management scenario of a single corporation, administrators would use a large number of managing nodes. We thus assume that 16 peers are sufficient for most actual management environments. Over the P2P management overlay we deployed up to 12 different NAC management services (namely, monitors for LDAP, DNS, DHCP, Radius, data base, Web servers, rogue user, firewall, proxy, access point, switches, and routers), in addition to the self-healing and configuration special services required in the recovery process. The single service policy enforced in all management services of our experiments defines that at least 2 instances per service must be running and, in case of failures, just 1 another instance must be activated per crashed instance.

Considering the above, two main sets of experiments have been carried out: multiple crashing peers, and variable number of peers and services. These experiments and associated results are presented in the next subsections.

4.1 Multiple Crashing Peers

The first experiment was designed to check the performance of the self-healing monitoring architecture when the number of simultaneously crashing peers hosting management services increases until the limit where half of them are broken. In addition, we want to check whether the number of instances of the self-healing and configuration services influences the recovery time and generated traffic.

For this set of experiments, we used to following setup: 12 management services are always deployed, each one with 2 instances running on the overlay. The total 24 service instances (*i.e.*, 12×2) are placed along 8 peers, each one thus hosting 3 (*i.e.*, $24 \div 8$) service instances. The number of crashing peers varies from 1 to 4. Since each peer hosts 3 instances, the number of crashing instances varies from 3 (12.5%) to 12 (50%), out of the total of 24 instances. Additional 4 peers have been used to host the self-healing and configuration services. Their varying number of instances has been organized, in pairs of self-healing/configuration, as follows: 2 and 4 instances, and 4 and 4 instances. Finally, we consider that group communication support is implemented interchangeably using multicast and unicast.

Figure 3 shows in seconds the time taken by the monitoring system to detect and activate new instances of the crashing services using the "spare" cluster nodes that host the configuration service. The first occurrence of 3 crashing services correspond to the situation where 1 peer fails; 6 crashing services correspond to 2 failing peers, and so on. No value is provide in 0 (zero) because with no failing peers there will not be any crashing service. Figure 4, in its turn, presents the network traffic generated by the management overlay in this recovery process. In this case, for 0 (zero) there exists an associated network traffic because, in the self-monitoring process, heartbeat messages are constantly sent regardless the presence or not of a failure.

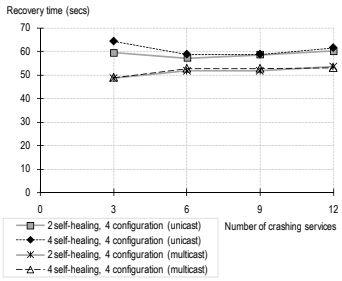


Fig. 3. Recovery time with multiple crashing

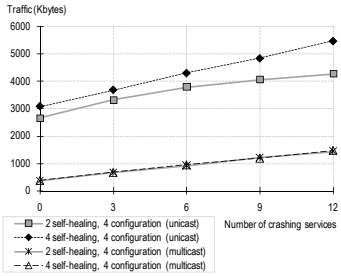


Fig. 4. Traffic to recover crashing peers

The recovery time as a function of the number of crashing peers stayed mostly constant. With that we can conclude that the system scales well considering a management scenario of 16 nodes. There is a little variance on the recovery time as a function of the

self-healing and configuration services. In fact, such difference is the result of employing multicast or unicast. When peers use multicasting they quickly become aware of changes in the system, and can rather react faster. Using unicast, however, more messages are sent, delaying the communication and, as a consequence, the reactions. In summary, the recovery time is not strongly influenced either by the self-healing and configuration services or by the number of crashing services. There is, however, a little influence from the use of multicast or unicast in the group communication support.

Network traffic, in its turn, presents a stronger influence of multicast or unicast support. As can be observed in Figure 4, multicast-based communications saves more bandwidth, which is expected. The important point to be observed, however, is that with the increasing number of crashed services the traffic generated to recover them is closely linear, but with doubling the number of failures, the traffic generate does not double together. Although not so efficient as in the case of recovery time, the bandwidth consumption is still scalable in this case. Putting these two parameters together and observing the graphs, if multicasting is used the number of self-healing and configuration services and the number of crashing peers do not influence the recovery time, and slightly increase the bandwidth consumption. In the case of unicast, however, the option of employing 2 self-healing instances instead of 4 is better, because this setup reacts slightly faster yet generating less traffic.

4.2 Varying Number of Peers and Services

The second experiment shows the relationship between recovery time and generated traffic when single crashes occur (which tends to be more frequent than multiple crashes) but the number of peers and services varies. We consider the recovery process when the number of management services increases (from 1 to 12, *i.e.* from 2 to 24 instances) over three setups where 2, 6, and 12 peers are used to host the management services. In addition to single crashes, we also fixed the number of 2 self-healing and 2 configuration services instances, hosted by 2 peers. We did so because, as observed before, the number of such instances few impacts on the recovery time.

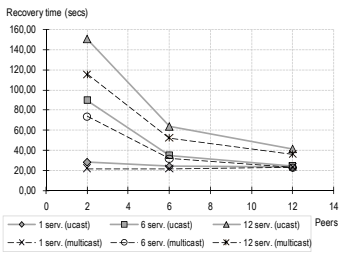


Fig. 5. Recovery time for multiple peers

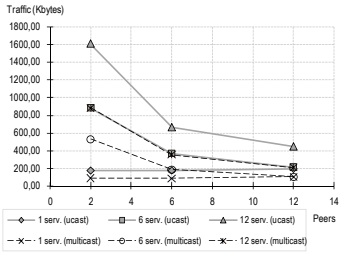


Fig. 6. Recovery traffic with multiple peers

In Figure 5, where the recovery delay is presented, services communicating via multicast are depicted with dashed lines, while services using unicast are depicted with solid gray lines. The recovery time when only 2 peers are employed is usually higher because each of the 2 peers hosts more service instances. When one of the peers crashes, more instances need to be activated. On the other extreme, with 12 peers, each peer hosts less services, leading to the situation where a crashing peer actually triggers the activation of less service instances.

The fact that more instances need to be activated as the result of a more load peer can be observed in Figure 6, that shows the traffic generated to recover the system. Again, multicast communications save more bandwidth than unicast, as expected. However, it is important to notice now that the number of services in each peer influences too. For example, 6 instances running on the same peer (line “6 serv. multicast”, with 2 peers in the x axis) despite being multicast still takes longer and generates more traffic to recover the system than the case where, via unicast, only 1 services is deployed (line “1 serv. unicast”, with 2 peers in the x axis).

This confirms that the number of peers and service instances must be similar in order to recover more promptly the system without generating too much traffic. If an administrator is restricted in terms of peers available, he/she must try to restrict the number of services employed as well. If new services are required, however, the option of also increasing the number of peers should be considered.

Now considering the whole picture, administrators should not worry about simultaneous crashes nor the number of self-healing and configuration services. Increased multiple crashes are more scare, and even if they happen the system is able to recover reasonably fast. As observed, the number of serf-healing and configuration services does not affect the overall performance of the system. However, administrator should do pay attention to the number of available peers and service instances, as mentioned before. Finally, the employment of multicast and unicast in the group communication mechanism influences in the recovery time (less) and the generated traffic (more). Choosing multicast whenever possible helps to improve the response time of the system. Unfortunately multicasting is not always available, which forces the administrator to use unicast to implement group communication.

5 Conclusions and Future Work

We have presented in this paper the design and evaluation of a P2P-based self-healing monitoring system employed in a NAC environment. The solution achieves self-healing capacity by splitting in two different processes the functions of failure detection and system recovery. Failure detection is executed inside management services that monitor final devices, while system recovery relies on special services called self-healing (that decides when new service instances must be activated) and configuration (that activates the new service instance as an reaction for the self-healing service decision).

The results of our experimental evaluations allow us to conclude that the number of instances of the self-healing and configuration service is not a major player in the performance of the system. They also permit us to state that simultaneously crashes on the management services does not influences so expressively the system performance

either. A network administrator willing to employ a self-healing monitoring solution should not concentrate his/her efforts in finding an ideal number of self-heling and configuration services. Our experiments employed 2 and 4 instances, respectively, and the system response was satisfactory. The fact that must be observed, however, is the group communication solution available on the managed network: multicast turns recovery faster while consuming less network bandwidth. Unfortunately, IP multicasting can not always be provided, and unicast ends up being chosen in such situations.

The most important aspects that must be observed in a self-healing monitoring solution is the number of peers employed in the P2P management overlay and the number of service instances deployed. With few instances, there is no need for several peers. On the other hand, with a large number of instances the number of peers should grow consistently, otherwise, on the occurrence of a failure, the recovery time will be higher and more network bandwidth is consumed by the intensive P2P traffic generated.

Currently, we are working on the optimization of the detection mechanism because the current version of it is responsible for a considerable amount of generated traffic. Another future work is the investigation about how service policies impact on the consumed network bandwidth and recovery time of our system.

Acknowledgement

Thanks to Dominique Dudkowski and Chiara Mingardi for contributing with this paper, and also to the Network and Support Division team at the Data Processing Center of UFRGS for the experience exchanged on the development of the UFRGS NAC system.

This work was partly funded by the Brazilian Ministry of Education (MEC/CAPES, PDEE Program, process 4436075), and by the European Union through the 4WARD project in the 7th Framework Programme. The views expressed in this paper are solely those of the authors and do not necessarily represent the views of their employers, the 4WARD project, or the Commission of the European Union.

References

1. Oetiker, T.: MRTG - The Multi Router Traffic Grapher. In: LISA '98: Proceedings of the 12th USENIX conference on System administration, Berkeley, CA, USA, USENIX Association (1998) 141–148
2. López, G., Cánovas, O., Gómez, A.F., Jiménez, J.D., Marín, R.: A network access control approach based on the AAA architecture and authorization attributes. J. Netw. Comput. Appl. **30**(3) (2007) 900–919
3. Perazolo, M.: A Self-Management Method for Cross-Analysis of Network and Application Problems. In: 2nd IEEE Workshop on Autonomic Communications and Network Management (ACNM 2008). (2008)
4. Trimintzios, P., Polychronakis, M., Papadogiannakis, A., Foukarakis, M., Markatos, E., Oslebo, A.: DiMAPI: An Application Programming Interface for Distributed Network Monitoring. In: Proceedings. 10th IEEE/IFIP Network Operations and Management Symposium, 2006. NOMS 2006. (2006) 382 – 393
5. Packard, H.: Management Software: HP OpenView (2008) Available in `http://www.openview.hp.com/`.

6. Agarwal, M.K.: Eigen Space Based Method for Detecting Faulty Nodes in Large Scale Enterprise Systems. In: IEEE/IFIP Network Operations and Management Symposium (NOMS 2008). (2008) CDROM.
7. Varga, P., Moldován, I.: Integration of Service-Level Monitoring with Fault Management for End-to-End Multi-Provider Ethernet Services. IEEE Transactions on Network and Service Management **4**(1) (2007) 28–38
8. Yalagandula, P., Sharma, P., Banerjee, S., Basu, S., Lee, S.J.: S3: a scalable sensing service for monitoring large networked systems. In: INM '06: Proceedings of the 2006 SIGCOMM Workshop on Internet Network Management, New York, USA, ACM Press (2006) 71–76
9. Prieto, A.G., Stadler, R.: A-GAP: An Adaptive Protocol for Continuous Network Monitoring with Accuracy Objectives. IEEE Transactions on Network and Service Management **4**(1) (2007) 2–12
10. Chaparadza, R., Coskun, H., Schieferdecker, I.: Addressing some challenges in autonomic monitoring in self-managing networks. In: 13th IEEE International Conference on Networks, 2005. (2005) 6 CDROM.
11. Zhou, Y., Lyu, M.R.: An Energy-Efficient Mechanism for Self-Monitoring Sensor Web. In: 2007 IEEE Aerospace Conference. (2007) 1–8
12. Granville, L.Z., da Rosa, D.M., Panisson, A., Melchiors, C., Almeida, M.J.B., Tarouco, L.M.R.: Managing Computer Networks Using Peer-to-Peer Technologies. IEEE Communications Magazine **43**(10) (2005) 62–68
13. Panisson, A., Melchiors, C., Granville, L.Z., Almeida, M.J.B., Tarouco, L.M.R.: Designing the Architecture of P2P-Based network Management Systems. In: Proceedings. IEEE Symposium on Computers and Communications (ISCC'06), Los Alamitos, CA, USA, IEEE Computer Society (2006) 69–75
14. Marquezan, C.C., dos Santos, C.R.P., Nobre, J.C., Almeida, M.J.B., Tarouco, L.M.R., Granville, L.Z.: Self-managed Services over a P2P-based Network Management Overlay. In: Proc. 2nd Latin American Autonomic Computing Symposium (LAACS 2007). (2007)
15. GPPD: Parallel and Distributed Processing Group – GPPD (2008) Available in `http://gppd.inf.ufrgs.br/new/`.

Self-managed services over a P2P-based Network Management Overlay

Clarissa Cassales Marquezan, Carlos Raniery Paula dos Santos, Jéferson Campos Nobre
Maria Janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco, Lisandro Zambenedetti Granville
Institute of Informatics – Universidade Federal do Rio Grande do Sul
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
Email: {clarissa,crpsantos,jcnobre,janilce,liane,granville}@inf.ufrgs.br

***Abstract*—The increasing complexity of IT systems requires sophisticated management solutions. Autonomic computing has been pointed as a possible solution for the management of this modern IT scenario, which includes the management of the underlying communication infrastructure. This paper focuses in how to provide proper network management in the light of complex IT scenarios. Since we believe that traditional network management approaches are not sufficient, one alternative that is rising as a potential solution is the employment of peer-to-peer (P2P)-based network management. Based on this scenario, we have proposed an architecture for building autonomic network management services on top of a P2P-based management overlays.**

I. Introduction

Network management is an important discipline whose one of its main goals is to maintain the IT communication infrastructures working in a proper manner. In the IT arena, some investigations already predict that in few years the complexity of IT systems will be such that even experienced system administrators [1] will not be able to properly address IT management issues. Since IT and networking tends to be more and more managed in an integrated fashion, we strongly believe that human network operators will face severe difficulties in managing future IT supporting networks. Even though several network management solutions have been widely proposed, implemented, and deployed, the question here is that such management solutions have been designed considering management scenarios different than those to be posed by the near future.

One alternative that is believed to have the potential to properly address this issue is the employment of *Autonomic Computing* (AC) features in Network Management Systems (NMS), thus creating Autonomic Network Management (ANM) systems. A generally accepted concept from the AC research is that an autonomic architecture is composed of *autonomic elements* whose communication may be provided by a distributed, service-oriented infrastructure [2]. Observing the established network management architectures [3] [4] [5], however, one can easily conclude that most (if not all) of them are not based on the service-oriented approach.

The recent researches on Web services for network management [6] have fortunately suggested the possibility of real service-oriented management systems. In addition, the introduction of P2P-based network management models [7] potentially enables highly distributed management solutions. Based on these two approaches, our research group has developed a new network management model [8]. Analyzing P2P management overlays, we believe that they form environments where autonomic computing principles could be employed in order to enable distributed, autonomic network management systems.

We propose in this paper an ANM architecture based on P2P overlays. This paper describes the architecture designed to maintain the network management services of a P2P-based NMS executing with self-healing, self-configuration, and self-protection features. The main contribution of this paper concerns with how we have explored the native features of P2P overlays in order to provide the autonomic features. The basic features explored were group communication, the native support for content distribution, self-organization support, and awareness about the elements that compose the P2P overlay. With the combination of P2P and autonomic computing is possible to define mechanisms to configure local peer and also remote peers without human explicit intervention, as well as, define self-healing algorithms to repair failures on the system.

The remainder of this paper is organized as follows. Section 2 surveys the related work. Section 3 brings the description of autonomic P2P-based network management architecture proposed in this paper. Finally, section 4 contains some conclusions and future work.

II. Related Works

There are few initiatives investigating the integration of P2P and autonomic computing features in order to solve the complexity problem of network management. There are attempts to build autonomic systems, but most of them are not directly devoted to network management. Generally, such efforts are designed to manage specific scenarios such as specialized servers [9] [10], grid and pervasive environments [11] [12] [13], wireless sensor networks [14] [15], or to provide more general frameworks [16] [17] [18]. On the other hand, there are a little more researches trying to merge network management systems with P2P features. This section presents some efforts that have been carried out to develop autonomic network management systems, P2P-based network management systems, and the convergence of ANM approach and P2P-based network management systems.

A. Autonomic Network Management Systems

One of the first proposed autonomic network management system is Focale [19]. Its architecture is built on top of a currently established network management environment. The solution adopted in Focale was the usage of information and data models, as well as ontologies, to provide self-knowledge for the autonomic system. The major strength of this proposal concerns with the fact that it does not require changes on already established management systems. However, this is also its drawback, since complex information and data model are necessary to describe the actions.

Another interesting approach is Service Clouds [20]. This environment intends to provide an infrastructure for service deployment in an autonomic fashion. It is not as generic as Focale, but it presents an autonomic alternative for maintaining communication channels where services are deployed. It brings the idea of autonomic services based on network overlays. The autonomic communication service is deployed inside the overlay and the communication service is able to self-adapt according to the network overlay conditions. Despite its interesting vision of autonomic services, Service Clouds are restrict to autonomic management of communication channels.

The policy-based approach is also explored by Morimoto et. al [21], Bahat et. al [22], and Meer et.al [23]. These systems employ a policy-based architecture to describe, in a high abstraction level, the overall behavior of the system, thus the autonomic elements of the network should maintain the policies inside the infrastructure. Another interesting work is presented by Ouda et.al [24], where autonomic features are used to allow dynamic changes not only on the managed resources but also in the policy itself.

B. P2P-based Network Management Systems

One of the first initiatives on developing P2P-based network management has been carried out by State et al. [25]. The authors proposed a Java Management Extensions (JMX)-encoded system based on the JXTA P2P framework [26]. Managed entities announced their management interfaces to remote management peers on the P2P network. Even though the authors discuss about the possible integration with “de facto” TPC/IP management solution, i.e., the Simple Network Management Protocol (SNMP) framework [27], the proposed architecture is not completely integrated with established management agents found inside current devices (i.e., router, switches, servers, etc.). This fact restricts the employment of this management system to a reduced set of resources.

The work of Binzenhöfer et al. [7] was designed employing P2P overlays to address fault and performance management. Their architecture aims at providing generic connectivity tests and QoS monitoring in a distributed and self-organized system composed of Distributed Network Agents (DNAs). The authors argue that their architecture can facilitate the construction of autonomic communication, since they provide QoS-enabling solutions and self-organization (based on DHTs). Although being a P2P-based management system and foreseeing manners to insert autonomic features, the system was not developed to naturally integrate P2P and autonomic computing principles in order to build a network management system.

The approach proposed by Kamienski et al. [28] merge traditional PBNM architecture with P2P architecture, resulting in a P2P Policy Management Infrastructure (P4MI). As an application of P4MI they had developed a PBM solution for Ambient Network, called PBMAN. PBMAN enables scalable mechanisms for network composition inside the AN, as well as policies distribution and retrieval. P4MI is composed of Policy Decision Networks (PDNs), Policy Enforcement Points (PEPs) and users agents. Through this approach it is possible to establish policies to manage devices or services. Until the present moment, according our awareness, there are no initiates to endow P4MI with autonomic features.

C. Autonomic and P2P-based network management systems

Ambient Network (AN) [29] is one of the first works that has called attention to the convergence of AC and P2P-based network management. The key concept behind AN is network composition, which means that the establishment of inter-network agreements must be performed on demand and without the intervention of human administrators during the process. The authors argue that an AN management system must be dynamic, distributed, and self-managed, since an AN itself is composed of hierarchical overlay. Thus, among other alternatives to build AN management systems, P2P-based management is one of the possible approaches used to maintain the hierarchy of the system by executing management and control tasks [30].

However, this AN P2P management approach does not consider, explicitly, well known and accepted features of autonomic computing. Indeed, it is possible to say that self-configuring could be achieved by the network composition mechanisms described in this paper. Thus, we believe that there are several aspects of P2P overlays that can be investigated in order to provide an autonomic network management system.

III. Autonomic P2P-based Network Management Proposal

The main objective of our approach is to provide a self-managed infrastructure to maintain the execution of services inside a P2P-based network management overlay. At this moment, our focus is to provide self-healing and self-configuration features, but we also address some self-protection aspects at some elements. In our vision, to achieve a self-optimized system it is necessary first to provide the former three autonomic features. Thus, based on what the system is aware of and based on its self-configuration capacity, it can predict the best choices to improve its performance and apply them. For this reason self-optimization is not addressed at this point of the research, but is a future work.

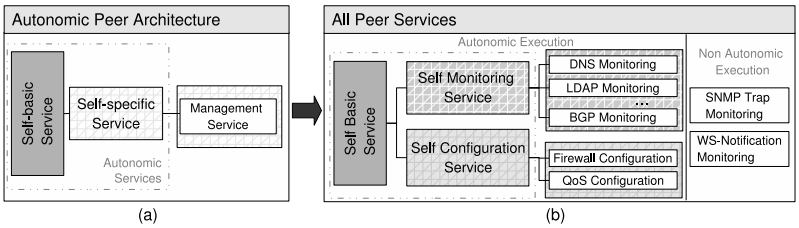


Fig. 1. Peer service architecture

A. Basic Principles

One of the lessons presented by Mortier and Kiciman [31] is the necessity of making explicitly assumptions about the operation environment when defining an autonomic control system. Thus, we made both generic and specific assumptions. The generic ones are related to the overall environment operation, and are listed below.

1) The P2P-based network management overlay must be able to operate based on service oriented approach. This means that the peers will provide and consume services deployed inside the P2P overlay;
2) The services inside the P2P overlay must be discovered somehow;
3) The P2P overlay must be able to deal with peer group interactions;
4) The overlay must be able to deal with the management of services with or without autonomic support. In order to build the autonomic support for the services, the developers will have to implement the set of operations we have defined;
5) The overlay services managed by the autonomic services must be as less conscious as possible about the fact that they are being managed;
6) The management of the overlay will be driven based on condition-action policies;
7) Our approach is designed using a timed asynchronous model [32] to address the fault tolerance aspects.

Based on these assumptions, we have defined different types of services, each one playing defined roles on our autonomic P2P-based network management architecture. Inside the overlay there are management services and autonomic services. Figure 1 (a) shows the peer service architecture and how these services are instantiated is presented in Figure 1 (b). The management services comprise the common network management services, such as DNS monitoring service, firewall configuration services, QoS configuration service, and so on. The autonomic services are designed to control the management services.

In our approach, there are at least two autonomic services: self-basic and self-specific, as presented in Figure 1 (a). The first one is named *Self Basic Service* and each peer, interested to provide autonomic features, must provide this service. It is responsible for managing the execution of the self-specific services. The second one, self-specific service, is targeted to control the management services of the overlay. Figure 1 (b) illustrates two self-specific services: *Self Monitoring Service* and *Self Configuration Service*. Each one manages different classes of management services. The former manages the monitoring service class and the latter manages the configuration service class. Each self-specific service can handle a set of overlay policies that must be applied for each management service or for the entire service class. Actually, the self-basic service is also controlled by a default, simple policy. The description of policy support and autonomic services is presented in the following sub-sections.

B. Policies

We have defined two types of possible policies: network and overlay. The former is treated by management services to control the network devices managed by the overlay system. One example of this type of policy is *Network policy A* presented in Figure 2. The latter is used with autonomic services in two situations: (i) management of one type of service inside the overlay service class, such as presented in the *Overlay policy A*; or (ii) management of all services from a service class, as *Overlay policy B* presents.

Network policy A

```
if (source_ip = 156.34.90.23) AND (destination_net = 143.54.12.0)
then
     mark DSCP 46
```

Overlay policy A	Overlay policy B
`if (service_class = monitoring) AND (service_type = DNS) then keep 3 DNS monitor services`	`if (service_class = monitoring) then keep 3 monitors for each service of the class`

Fig. 2. Policy examples

The major idea behind our policy support is the combination of policy-based network management (PBNM) [5] with P2P concepts. Traditional PBNM architecture comprises four entities: policy tool, policy repository, policy decision point (PDP), and policy enforcement point (PEP). We redefined these entities considering explicitly P2P features. The major changes are related to the manner that policy repository,

PDP, and PEP entities are treated. These entities became management services inside the overlay.

We defined a management service called *Policy Store* which is responsible for storing all type of policies. This service must be running in several peers in order to form a peer group that share the same policy information. This restriction allows us to build a decentralized policy repository. This service provides operations to receive policies and also to provide the requested policies. There are several aspects comprehending our policy-based model that will not be addressed at this moment, because this is not the focus of this paper.

At this moment, we are interested in showing the relationship between the overlay policy actions and the operations supported by the autonomic services. This relation is expressed by an algorithm that describes the required steps to implement the action. This algorithm is implemented by *Self Basic* and self-specific services. So far, we have defined the following overlay policies actions: `keep`, `use`, `instantiate` and `kill`. However, as the system evolves it is possible to build more sophisticated overlay policy actions. But, in this paper, we will just present the algorithms to provide `instantiate` action, implemented by the *Self Basic* service; and `keep` action implemented by the self-specific service.

C. Self Basic service

Every peer of the P2P overlay must provide the *Self Basic* service if the management services of this peer are supposed to be controlled in an autonomic fashion. This service is responsible for the self-configuration feature on our model, hence its major task is to deal with the initialization of self-specific services. This task is represented in the default policy of the *Self Basic* service. Figure 3 presents the default policy and its associated algorithm.

This autonomic service does not need to search for its policy in the *Store Policy* service. The default policy is already present since the service initialize its execution, and is illustrated in Figure 3 (a). The algorithm that specifies this policy comprises two moments. The first one is related to the initialization of self-specific services of the peer, while the second one regards to the maintenance of the self-specific services during the life time of the peer.

Figure 3 (b) presents the first part of algorithm. It shows that just necessary self-specific services are instantiated, i.e., even if there are management services of different classes, inside the peer the algorithm will initialize self-specific services only for management services with autonomic support. This restriction avoids wasting resources with services that are not required.

Figure 3 (c) shows the second part of the algorithm. The *Self Basic* service is able to identify whether some new management service was introduced inside the local peer, and according to the capacities of this service, it will be controlled by some self-specific service already running inside the peer or the *Self Basic* service will instantiate the appropriated autonomic service.

Another feature addressed in the second part of this algorithm is the ability of solving local failures on the self-specific

(a) Self Basic default policy

```
if management_service with
        autonomic support
then
     instantiate management_service
```

(b) Self Basic service tasks on peer start up

```
01. for each management_service
02.     if there is autonomic support
03.     then
04.         if there is not a self_specific_service executing
05.         then
06.             instantiate the self-specif_service
```

(c) Self Basic service tasks during the live time of the peer

```
01. while peer executing
02.     check Presence_service events for this default_policy
03.     if new management_service provided
04.     then
02.         if there is autonomic support
03.         then
04.             if there is not a self_specific_service executing
05.             then
06.                 instantiate the self-specific_service
07.     if some self_specific_service failed
08.     then
09.         re-instantiate the self_specific_service
10.         if re-instantiation failed
11.         then
12.             for each management_service controlled by
                                self_specific_service in failure
13.                 check available peer
14.                 if there is some peer available
15.                 then
16.                     instantiate the management_service in remote peer
17.                 else
18.                     kill management_service
19.                     wait for the contact of the peer group service in failure
```

Fig. 3. Default policy and its algorithm for *Self Basic* service

services. On a first moment, the *Self Basic* service tries to handle the problem locally. However if it is not able to re-instantiate the service in failure or cannot find another peer to re-instantiate the orphan management services, then it waits for the contact of the peer group of the self-specific service. When the peer group contacts it, then it informs that the peer group will have to relax their restrictions involving the policies associated to the management services of the self-specific service in failure. In order to accomplish this negotiation we have developed a restriction policy relaxation protocol. This protocol is inspired in the same kind of mechanism found in negotiation protocols from multi-agent research area, where the agents must obtain an agreement on which resources they can or cannot release [33]. Our current restriction policy relaxation protocol defines that the system will keep executing despite of the occurred problem, but in background it will continuously try to find another peer to re-instantiated the orphan services.

It is important to notice that in this failure case, we introduced some self-protection features. When the *Self basic* service is not able to re-instantiate the orphan management services, it will kill the local ones and the responsibility to find a solution is transferred to the self-specific peer group. With this restriction we can prevent malicious peers that are

trying to damage the overlay execution, or the ones that try to attack the devices controlled by the management services. In the first case, without killing the management services, the overlay can be flooded with many unnecessary management services, hence this will introduce communication and processing overhead in the peers of the overlay. On the second case, if the management services were not killed, they will keep communicating and executing their management tasks over the devices, and this can cause deny of service attacks or bizantine failures on the devices.

Other relevant feature of *Self Basic* service concerns with fact that it does not belong to any peer group. We decided to impose this behavior because their tasks are restricted to local control of self-specific services. On the other hand, self-specific service is targeted to form peer groups in order to control the management services spread along the overlay. These autonomic services are presented in next sub-sections.

D. Self-specific service

The adjective *specific* in the self-specific service name is related to the idea of different classes of services inside the overlay. In Figure 1 (b), we present two classes of services: monitoring, and configuration. The idea behind the specialization of self-specific services, is allowing the definition of peer groups, whose main objective will be maintaining the policies applied to the service class. Using the concept of peer groups, it is possible to restrict the number of peers inside the overlay that will receive the messages related to a policy overlay action. Thus, considering the example in Figure 1 (b), the self-monitoring service will not receive the control messages related to the policy overlay actions of configuration service class.

One of the responsibilities of a self-specific service is to retrieve the policies associated with the service class. This element will contact the *Policy Storage* service and handle the appliance of the policy. For each type of service, there could be different policies. We have defined that one of the policy attributes will be its appliance priority. Based on this information, the self-specific service will be able to choose which policy must be first deployed. We are aware that there are conflict problems involved in this kind of solution. Considering the scope of the policy, this conflict can be solved internally by the self-specific service of the peer. However, if there is a peer group managing the same set of policies, this conflict must be solved by the entire peer group. In order to solve this problem, we also use the restriction policy relaxation protocol.

Another responsibility of this service is the maintenance of the policies, which means that it must implement each algorithm associated with the overlay policy actions it must ensure. As the assumptions made in *Self Basic* service, we assume that the implementation of the supported overlay policy actions is already placed inside the peer. Another assumption we have made regards the relationship between self-specific services and the management services they control. So far, we assume that each peer that has management services with

```
01. while keep_policy applied
02.     check presence_service events for this keep_policy
03.     if local management_service down
04.     then                                              Local Peer Actions
05.             instantiate new management_service
06.             if  instantiation failed
07.             then
08.                     call solve_local_failure
09.     if some self_specific_service down
10.     then                                              Peer Group Actions
11.             elect one self_specific_service as leader
12.             if this is the self_specific_service leader
13.             then
14.                     verify if the peer containing the failed self_specific_service has crashed
15.                     if peer has crashed
16.                     then
17.                             call solve_peer_group_inconsistency
18.                     else
19.                             contact self_basic_service of failed self_specific_service peer
20.                             if self_basic_service is not answering
21.                             then
22.                                     call solve_peer_group_inconsistency
23.                             else
24.                                     call restriction policy relaxation protocol
```

(a)

```
i.  solve_local_failure {
ii.  check available peer
iii. if there is some peer available
iv.  then
v.       instantiate management_service
           in remote peer
vi.      if there is no other management_service
           for local self_specific_service
vii.     then
viii.            self_specific_service kill itself
ix.      else
x.               call restriction policy
                      relaxation protocol
xiii. }
```

(b)

```
i.  solve_peer_group_inconsistency{
ii.  check available peer
iii. if there is some peer available
iv.  then
v.       instantiate management_service
           in remote peer
vi.  else
vii.     call restriction policy relaxation protocol
viii. }
```

(c)

Fig. 4. Algorithm for the implementation of keep action

autonomic support will also have a self-specific service. This assumption restrict the capacity of self-specific services to control management services from remote peers.

As mentioned before, one relevant aspect related to self-specific service is its direct relationship with the policy action. For example, consider the following scenario: there is just one peer with the *Self Monitoring* service and the *DNS Monitoring* service (Figure 1 (b)) inside the network management overlay, and this overlay service downloads *Overlay Policy A* (Figure 2) from the *Policy Storage* service. It means that inside the overlay there will must be at least 3 peers that have the *DNS Monitoring* service. Based on the assumption described previously, and on the *Overlay Policy A*, we must instantiate two more *Self Monitoring* services to handle the two extra *DNS Monitoring* services inside some peers of the overlay. These *Self Monitoring* services will have to form a peer group where they will exchange information in order to maintain the policy. In order to achieve this goal, self-healing and self-configuration features must be defined by the keep algorithm. Figure 4 presents the algorithm written in pseudo code.

As presented in Figure 4 (a), the self-specific service can handle problems locally or together with the peer group. In both cases, the self-healing feature is achieved based on the information retrieved from the *Presence* service. This service, basically, follows a publish/subscribe model, where an interested user subscribes in a service what information he wants to be notified and the service, based on the information generated

by an element (i.e., person, device or software), sends the correct presence information. The definition of autonomic requirements for *Presence* service enables the detection and diagnosis of local and peer group problems. According to this information the `keep` action algorithm is able to repair the problem.

It is possible to notice that the repair procedures implemented by this algorithm are based on *solve_local_failure* (Figure 4 (b)) and *solve_peer_group_inconsistency* (Figure 4 (c)) methods. When the problem must be solved by the peer group, it is first necessary to decide which peer of the group will drive the situation. For this reason, they have to elect the responsible peer, which tries to verify if the peer of the self-specific service in failure has totally crashed or if at least the *Self Basic* service is still running. Based on this verification, the elected peer can decide which actions it will take to heal the overlay service.

Summarizing, we can say that *Self Basic* service provides self-configuration features to our architecture, while the self-specific services provide the self-healing feature. As a future work, we intend to introduce self-optimization features based on what the system could learn from the behavior of the autonomic services already defined.

IV. Conclusion

The complexity of IT systems is a well discussed problem and it is not an isolated phenomenon. Besides the complexity of IT systems, there is also the complexity of the underlying networks that provide support for IT communications. A proposal that is rising as a potential solution for complexity problem is the utilization of autonomic computing (AC) features. At this meanwhile, in the network management scenario, the P2P-based network management is a promising solution for achieving scalability, connectivity, and fault tolerance features. However, the complexity problem is still a issue not addressed yet.

Thus, considering current trends and rising solutions, we have proposed an autonomic network management system based on P2P overlays. The major advantage of using a P2P approach to build an autonomic management architecture is related to fact that we do not need to care about the network topology of the system. We can use the self-organizing, connectivity, and self-discovering features of the P2P model to get the basic infrastructure of the autonomic system executing in a total distributed fashion. In a P2P-based approach, there is no need of brokers or any other kind of central entities.

Our main objective is handling the behavior of P2P overlay management services in an autonomic fashion. In order to define this architecture, we have done several assumptions. The generic ones drive the definition of the autonomic P2P-based environment, and they describe the expected behavior of P2P overlay elements. These elements are services provided by the overlay peers. We have defined two autonomic services. The former is *Self Basic* service that is designed to provide self-configuration features of the architecture. The latter is self-specific service, which is responsible for self-healing aspects.

Although we have defined self-configuration, and self-healing features in our model, these features can be improved. In order to improve the self-configuration aspect, we intend to develop two new overlay services. The first one will be responsible for storing and delivering the self-specific implementations to the peers that does not provide them. This service will be accessed mainly by the *Self Basic* service The second one will store and deliver the policy action implementations to the self-specific services that do not have support for such implementation. The self-healing support will be better explored improving the restriction policy relaxation protocol. We are researching negotiation mechanisms that do not introduce too much communication and processing costs for the overlay, but that also can better treat the interesting conflicts of each policy.

Moreover, as a future work, we intend to implement these architecture using different P2P technologies. Currently, this architecture is implemented based on JXTA framework and the system is named Autonomic ManP2P (AManP2P). We also intend to make performance evaluations concerning the bootstrapping process, reaction time in case of failure, and the overhead introduced by the protocols that maintain the autonomic management of the services inside the overlay.

References

[1] J. O. Kephart, "Research challenges of autonomic computing," in *ICSE '05: Proceedings of the 27th international conference on Software engineering*. New York, NY, USA: ACM Press, 2005, pp. 15–22.

[2] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 32, no. 1, pp. 41–50, 2003.

[3] G. Goldszmidt and Y. Yemini, "Distributed Management by Delegation," in *Proceedings of the 15th International Conference on Distributed Computing Systems, 1995*, Vancouver, BC, Canada, May-June 1995, pp. 333–340.

[4] R. Stephan, P. Ray, and N. Paramesh, "Network management platform based on mobile agents," *International Journal of Network Management*, vol. 14, no. 1, pp. 59–73, 2004.

[5] J. Strassner, *Policy-Based Network Management: Solutions for the Next Generation*, R. Adams, Ed. San Francisco: Morgan Kaufmann, August 2003.

[6] R. L. Vianna, E. R. Polina, C. C. Marquezan, L. Bertholdo, L. M. R. Tarouco, M. J. B. Almeida, and L. Z. Granville, "An evaluation of service composition technologies applied to network management," in *Proceeding of 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, May 2007, pp. 139–148.

[7] A. Binzenhöfer, K. Tutschku, B. auf dem Grabem, M. Fiedler, and P. Carlsson, "A P2P-Based Framework for Distributed Network Management," in *Proceedings. Wireless Systems and Network Architectures in Next Generation Internet*, ser. Lecture Notes in Computer Science, vol. 3883. Heidelberg, Springer-Berlim, 2006, pp. 198–210.

[8] L. Z. Granville, D. M. da Rosa, A. Panisson, C. Melchiors, M. J. B. Almeida, and L. M. R. Tarouco, "Managing Computer Networks Using Peer-to-Peer Technologies," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 62–68, 2005.

[9] T. White, D. Calvert, and J. Litkey, "Design of an Autonomic Element for Server Management," in *Proceedings. Second International Conference on Autonomic Computing, 2005. ICAC 2005.* IEEE Press, june 2005, pp. 147–158.

[10] D. M. Chess, G. Pacifici, M. Spreitzer, M. Steinder, A. Tantawi, and I. Whalley, "Experience with Collaborating Managers: Node Group Manager and Provisioning Manager," in *Proceedings. Second International Conference on Autonomic Computing, 2005. ICAC 2005.* IEEE Press, june 2005, pp. 39–50.

[11] A. J. Chakravarti, G. Baumgartner, and M. Lauria, "The organic grid: self-organizing computation on a peer-to-peer network," in *Proceedings. International Conference on Autonomic Computing, 2004. ICAC 2004.* IEEE Press, may 2004, pp. 96–103.

[12] D. Lewis, T. O'Donnell, K. Feeney, A. Brady, and V. Wade, "Managing user-centric adaptive services for pervasive computing," in *Proceedings. International Conference on Autonomic Computing, 2004. ICAC 2004.* IEEE Press, may 2004, pp. 248–255.

[13] T. Guan and E. Zaluska, "An autonomic service discovery mechanism to support pervasive device accessing semantic grid," in *Proceedings of The 4th IEEE International Conference on Autonomic Computing (ICAC2007)*, Jacksonville, Florida, USA, 2007.

[14] T. R. M. Braga, F. A. Silva, L. B. Ruiz, J. M. S. Nogueira, and A. A. F. Loureiro, "Design and Evaluation of an Autonomic Sensor Element," in *Proceedings. First Latin American Autonomic Computing Symposium, 2006. LAACS 2006.*, july 2006, pp. 36 –47.

[15] T. Braga, F. Silva, L. Ruiz, J.-M. Nogueira, and A. A. Loureiro, "A tiny and light-weight autonomic element for wireless sensor networks," in *Proceedings of The 4th IEEE International Conference on Autonomic Computing (ICAC2007)*, Jacksonville, Florida, USA, 2007.

[16] H. Liu and M. Parashar, "A Component Based Programming Framework for Autonomic Applications," in *Proceedings. International Conference on Autonomic Computing, 2004. ICAC 2004.* IEEE Press, may 2004, pp. 10–17.

[17] E. Patouni and N. Alonistioti, "A framework for the deployment of self-managing and self-configuring components in autonomic environments," in *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 480–484.

[18] M. Steinder, I. Whalley, D. Carrera, I. Gaweda, and D. Chess, "Server virtualization in autonomic management of heterogeneous workloads," in *Proceeding of 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, May 2007, pp. 139–148.

[19] J. C. Strassner, N. Agoulmine, and E. Lehtihet, "Focale - a novel autonomic network architecture," in *LAACS 2006: Proceedings of the 1st Latin American Autonomic Computing Symposium*, 2006, pp. 48–60.

[20] P. k. Mckinley, F. A. Samimi, J. K. Shapiro, and C. Tang, "Service clouds: A distributed infrastructure ofr constructing autonomic communication services," in *DASC'06: Proceeding of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, 2006.

[21] K. Morimoto, M. Matsubara, and K. Suzuki, "Policy-based autonomous management system," in *LAACS 2006: Proceedings of the 1st Latin American Autonomic Computing Symposium*, 2006, pp. 70–81.

[22] R. M. Bahat, M. A. Bauer, E. M. Vieira, and O. K. Baek, "Using policies to drive autonomic management," in *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 475–479.

[23] S. van der Meer, A. Davy, S. Davy, R. Carroll, B. Jennings, and J. Strassner, "Autonomic networking: Prototype implementation of the policy continuum," in *Proceedings. The 1st International Workshop on Broadband Convergence Networks, 2006. BcN 2006.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 1–10.

[24] M. B. Abdelnasser Ouda, Hanan Lutfiyya, "Towards automating the adaptation of management systems to changes in policies," in *Proceedings. 10th IEEE/IFIP Network Operations and Management Symposium, 2006. NOMS 2006.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 1–4.

[25] R. State and O. Festor, "A Management Platform Over Peer-to-Peer Service Infrastructure," in *Proceeding. 10th International Conference on Telecommunications, 2003. ICT 2003*, Vancouver, BC, Canada, 2003, pp. 124–131.

[26] L. Gong, "JXTA: A Network Programming Environment," *IEEE Communications Magazine*, vol. 5, no. 3, pp. 88–95, May 2005.

[27] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," Internet Engineering Task Force (IETF), RFC 3411, STD 62, Dec. 2002.

[28] C. Kamienski, D. Sadok, J. F. Fidalgo, and J. Lima, "On the Use of Peer-to-Peer Architectures for the Management of Highly Dynamic Environments," in *Proceedings. Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops 2006)*. IEEE Press, march 2006, 1 CD-ROM.

[29] M. Brunner, A. Galis, L. Cheng, J. A. Colás, B. Ahlgren, A. Gunnar, H. Abrahamsson, R. Szabo, S. Csaba, J. Nielsen, A. G. Prieto, R. Stadler, and G. Molnar, "Towards Ambient Networks Management," in *Proceedings. Mobility Aware Technologies and Applications*, ser. Lecture Notes in Computer Science, vol. 3744. Heidelberg, Springer-Berlim, 2005, pp. 215–229.

[30] C. Simon, R. Szabo, P. Kersch, B. Kovács, A. Galis, and L. Cheng, "Peer-to-Peer management in Ambient Networks," in *Proceedings. IST Mobile and Wireless Communications Summit 2006*, 2006.

[31] R. Mortier and E. Kiciman, "Autonomic network management: some pragmatic considerations," in *INM '06: Proceedings of the 2006 SIGCOMM Workshop on Internet Network Management*. New York, NY, USA: ACM Press, 2006, pp. 89–93.

[32] F. Cristian and C. Fetzer, "The timed asynchronous distributed system model," *IEEE Transactions on Parallel and Distributed Systems*, pp. 603–618, 1999.

[33] F. Hassan and D. Robertson, "Constraint relaxation to reduce brittleness of distributed agent protocols," in *16th European Conference on Artificial Intelligence (ECAI' 04)*, May 2004.

Performance Evaluation of Notifications in a Web Services and P2P-Based Network Management Overlay

Clarissa Cassales Marquezan, Carlos Raniery Paula dos Santos, Ewerton Monteiro Salvador,
Maria Janilce Bosquiroli Almeida, Sérgio Luis Cechin, Lisandro Zambenedetti Granville
Institute of Informatics – Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 – Porto Alegre, RS – Brazil
Email: {clarissa, crpsantos, emsalvador, janilce, cechin, granville}@inf.ufrgs.br

***Abstract*—Notification service on network management is an essential tool that helps to save network resources, such as bandwidth. A management notification is basically an event message that reports a resource's internal state to an interested manager. A complete notification support is not accomplished via simple tasks anymore, for example, restricted to notifying managers from the same administrative domain of the managed devices. Due to huge changes on current network usage, this management topic needs to be reviewed in the light of modern technologies and requirements. In this paper we present a notification service integrated in a P2P-based network management solution called ManP2P. We also present an experimental evaluation regarding propagation delay and processing costs. We believe that with our solution it is possible to enhance P2P-based network management advantages without paying a great performance cost.**

I. Introduction

Network management is a critical discipline to support the IT infrastructure of modern organizations. Often, organizations with a poorly managed network eventually face problems associated to the provisioning of their final services, which may easily evolve to financial losses too. This leads to the essential observation that having a healthy network is crucial for the organizations' businesses.

Performance has always been a key issue of management solutions. Several mechanisms have been defined to improve the performance of management processes, or to reduce the impact of a management protocol over the managed resources. In this paper we will concentrate on investigating one of such mechanisms: management notifications. Notifications are essentially an event message issued by a managed resources towards one or more interested managers that describes the resource's internal status. Notifications can be used, for example, to report a device failure or an attack attempt.

Early key network management technologies have been defined in the 1980's, when the Internet Engineering Task Force (IETF) standardized the first TCP/IP management solutions. One of the main results from that effort was the Simple Network Management Protocol (SNMP) [1], which ended up becoming the *de facto* management standard in the computer networking industry. Nowadays, however, one cannot plan the management of a computer network solely considering traditional technologies and the managed network itself; there is also the necessity of taking into account all remaining systems that run on top of the managed network, i.e., the management of both network and systems need to be carried out in an integrated fashion. Unfortunately, SNMP does not consider this important integration, which leads to the situation where SNMP is seen as an obsolete protocol in several modern IT scenarios.

Recently, two other important technologies have been capturing the industry and academia attention as alternatives for integrated network management: Web services (WS) and peer-to-peer (P2P). WS [2] is a general solution to enable the communication among processes located along the Web by using eXtensible Markup Language (XML) documents transported on top over widely deployed Internet protocols such as HTTP, SMTP, and FTP. WS specifically designed for network management have been investigated in academia since 2003, with relevant results being published so far [3]. In the industry, two important efforts specify WS standards for services management (which encompasses network management as well): OASIS Management Using Web Services (MUWS) [4], and DMTF Web Services Management (WSM) [5]. These two efforts are still active and some reports indicate that they tend to converge in a single standard in the near future.

P2P for management [6], on its turn, started to be investigated more recently, in 2005 [7]. The general assumption is that P2P typical characteristics such as distributions, scalability, and reliability could "naturally" enhance the quality of current network management solutions. One important advantage of using P2P technologies for management is that a management overlay, potentially crossing different administrative domains, can be dynamically formed integrating human administrators that share a common goal and, most importantly, integrating diverse systems and networks that need to be managed in an integrated manner, as mentioned before.

It is not reasonable to assume, however, that WS and/or P2P would immediately become replacements to SNMP. SNMP is a very light protocol that consumes few network resources (e.g., bandwidth) if compared with the verbose XML-based

protocols employed by WS, and requires very few control when compared with the P2P control messages needed to maintain the P2P overlay. Being lighter, the processing of SNMP messages is supposedly faster than the processing of XML-encoded messages. In summary, the performance of WS and P2P tends to be worse than the performance of SNMP, despite their advantages listed before.

In SNMP, notifications (the focus of this paper) are materialized through the *trap* message. WS communications, in turn, are based on the Simple Object Access Protocol (SOAP) [8] and WS notifications can be implemented using either "plain" SOAP messages or specific notification standards such as WS-Event [9] or WS-Notification [10]. Notifications in P2P systems can be implemented using native P2P protocols, such as those available in the JXTA P2P framework [11]. In this paper we investigate the use of WS and P2P technologies for management notifications through the introduction of a management overlay named ManP2P. Over this P2P overlay, management notifications are delivered to interested managers using "plain" SOAP messages running on top of a JXTA infrastructure. Final managed systems (e.g., network devices) are integrated into the management infrastructure through the employment of intermediate gateway peers (called mid-level managers), whose main responsibility is to translated SNMP traps to P2P notification, i.e., SNMP is not replaced but instead integrated.

The main contribution of this paper is that it presents the advantages and drawbacks of using WS over P2P for management notifications considering a real network management system. We believe that our results can guide future decisions of network administrators aiming at employing WS and/or P2P in their management environments as either substitutes or complements of SNMP. To present our research, the remainder of the paper is organized as follows. Section 2 review P2P-based management in the light of notifications necessity. In Section 3 we present our ManP2P management overlay, used to perform the evaluation experiments. These experiments, and associated results and analysis, are discussed in Section 4. Finally, the paper is concluded in Section 5, where final remarks and future work are presented.

II. Notifications in WS and P2P-Based Management

The general assumption behind using P2P technologies for network management is that typical P2P advantages could be incorporated into the management discipline. Since P2P generally provides enhanced connectivity, better scalability, self-organization, and possible support of fault tolerance [12], bringing these features to the management of modern environment is clearly of great interest.

One of the first investigations on P2P-based management has been carried out by State *et al.* [6]. The authors proposed a Java Management Extensions (JMX)-encoded system based on the JXTA P2P framework [11]. Managed entities announced their management interfaces to remote management peers on the P2P network. Even though the authors discuss about the possible integration with the SNMP framework [13], very few considerations about integrating SNMP notifications (i.e., traps) into the management solutions has been made.

Binzenhöfer *et al.* [14], in turn, designed a P2P overlay to address fault and performance management. Their architecture targeted to provide generic connectivity tests and QoS monitoring in a distributed and self-organized system composed of Distributed Network Agents (DNAs). Although the architecture is a P2P-based one, notification support has not been addressed.

Brunner *et al.* [15] has introduced the use of P2P for the management of Ambient Networks (ANs). The authors argue that an AN management system must be dynamic, distributed, and self-managed. The P2P-based management is then one of the possible approaches used to maintain the hierarchy of an AN system by executing management and control tasks. AN elements are able to form Ambient Virtual Pipes (AVPs), which expose a management service overlay channel. Through AVP and AN composition one can provide management services for different domains. In fact, this approach realizes the management at a very high abstract layer, leading to a situation where the management of devices and basic services are not easily accomplished.

Granville *et al.* have presented in 2005 [7] a P2P management overlay called ManP2P. The initial version of our solution addressed three main management aspects: human-centered cooperation for management, increased connectivity between management entities, and balancing the management load using groups of peers. The authors have checked the P2P traffic required to retrieve a router's routing table, but at that time no considerations about notification support has been provided.

Although not always connected to P2P-based management, the use o WS for notifications has been presenting its own development. The WS-Event [9] and WS-Notification [10] specifications, for example, define SOAP messages tailored to transport notifications from one WS-enabled process to another considering the Internet as the underlying communication infrastructure. Although WS are independent of P2P, we believe that WS notifications running over P2P overlays is a quite interesting solution for today's needs: WS provide a widely accepted, supported, and employed protocol (i.e., SOAP), while P2P enables sophisticated communications on the Web that are impossible to be provided by traditional client-server protocols, such as HTTP.

The investigation work carried out up to today by the research community have mainly focused on introducing P2P-based management, but notifications have typically neglected to a second plane. Integration with SNMP has been addressed at some extend, but usually limited to delay and bandwidth consumption observations, and rarely considering SNMP traps. In a previous stage of our own research, we have initially evaluated the P2P support for notifications checking some scalability, bandwidth consumption, and response time aspects [**?**]. The work to be presented in the next sections of this paper, however, complements that very initial evaluation presenting

important additional and more conclusive results. Initially we present the ManP2P management overlay, to then evaluate the notification support through a set of experiments.

III. Notification Support in the ManP2P Management Overlay

In this section we introduce the ManP2P management overlay initially focusing on the entities that may mostly affect the performance of the notification support, i.e., focusing on so called top-level and mid-level managers. We also present some architectural details of these ManP2P entities that implement a publish-subscribe notification mechanism. Finally, the protocol stacks required to integrate SNMP-enabled devices in the ManP2P overlay are discussed. These stacks are important because they also affect the performance of the notification support.

A. Top-Level and Mid-Level Managers

Top-level and mid-level managers are entities already in use by several network management solutions, some of them in research projects or even in commercial products [16]. In ManP2P, top-level and mid-level managers additionally incorporate new functions and became dual-role entities: they are managers when controlling the system to be managed, but they are also communicating peers of the management overlay.

Top-level managers (TLMs) are management peers that implement the management front-end used by human network administrators. Through a TLM's graphical user interface (GUI) a network administrator can, for instance, contact other administrators located in remote domains, request the execution of management tasks to mid-level managers, and receive and visualize management notifications forwarded by mid-level managers. This last feature is of special interest in this paper. Figure 1 presents a snapshot of the ManP2P GUI highlighting the reception of notifications.

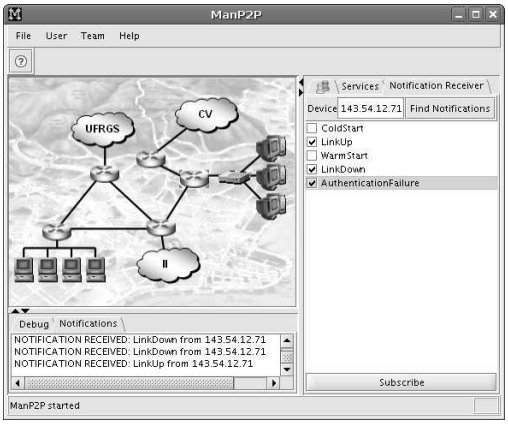


Fig. 1. ManP2P front-end snapshot

Mid-level managers (MLMs), different from TLMs, present no GUI to the network administrators. A (MLM) is solely an auxiliary management peer that exports management services to the management overlay and forwards device notifications to TLMs. In this last case, a MLM can be seen as a protocol gateway that translates SNMP trap messages to SOAP-based notifications delivered through the management overlay.

A key concept in ManP2P is of **management service**. A management service is an abstract entity hosted by MLMs that offers management functions to remote peers (TLMs and MLMs). One single MLM can host several management services at the same time, and one single management service can be implemented by several MLMs belonging to the same peer group. In this last case, the service is available to the management overlay while at least one MLM is present in the group. This strategy increases the availability of services by including new MLMs in a peer group that exposes a critical management service. In addition, the employment of peer groups allows balancing the management load by distributing service calls among the MLMs of a peer group. The notification support of ManP2P is implemented as a set of management services.

B. Services for Notification Support

The ManP2P notification support is based on the publish-subscribe model [17], where TLMs interested in specific notifications subscribe to the management overlay to receive the notifications they are interested in. In fact, a subscription request is sent to a MLM that will, in the future, forward the notifications back to the TLM. In order to discover which MLM the subscribing TLM must contact, a P2P discovery process is triggered using the JXTA discovery support. Since the details of this discovery process is out of the scope of this paper, we will concentrate only on the subscription actions performed after the TLM has found the MLM to be contacted. The ManP2P components that support the publish-subscribe model are presented in Figure 2.

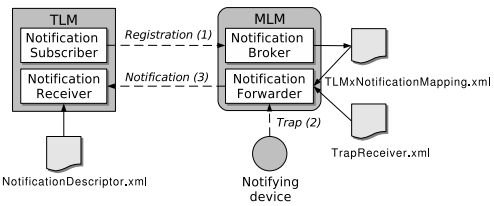


Fig. 2. TLM and MLM architecture

At the TLM, the *notification subscriber* component request its subscription to the remote *notification broker* component at the MLM. All successful subscription requests are also logged in the TLM `NotificationDescriptior.xml`

file for future use, for example, when the TLM re-initiates and wants to subscribe to remote MLMs again. At the MLM side, the notification broker stores the subscribing TLM identity in the `TLMxNotificationMapping.xml` file, which is accessed when forwarding new SNMP traps to interested TLMs.

The *notification forwarder* is the MLM component responsible for receiving SNMP traps and forwarding them to the interested TLMs. Since this component works as a trap receiver on its own, some basic SNMP configurations are required, such as the UDP port used to listen for new traps, as well as the IP address used in the networking interface, which may happen to be a regular unicast address as well as a multicast one (more details in the next subsection). This configuration information is stored in the `TrapReceiver.xml` file, which is read by the *notification forwarder* in its initialization.

Once an SNMP trap is received by MLM, the *notification forwarder* reads the `TLMxNoticiationMapping.xml` file in order to identify the interested TLMs. For each TLM, a SOAP notification message is forwarded and received by the `notification receiver` component at the TLM side. The actions performed by the TLM in reaction to the just received notification depends on the other TLM components, who are independent of the notification mechanism.

It is important to highlight that all these publish-subscribe components are implemented as management services accessed by SOAP messages transported over the JXTA P2P overlay. Each notification itself is a SOAP message too, as mentioned before. Although specific definitions for SOAP-based notification exist (e.g., WS-Event and WS-Notification), we do not employ these definitions in our evaluation because the current implementation of the JXTA-SOAP framework used in this work implements neither WS-Event nor WS-Notification, which forces the use of "plan" SOAP messages in all communications.

C. Protocols for Notification Transport

Our implementation of ManP2P is based on two main frameworks mentioned before: JXTA [11] and JXTA-SOAP [18]. JXTA is a popular open source and generic framework to build P2P solutions. JXTA-SOAP is an implementation of SOAP [8] over the JXTA P2P infrastructure, which enables the development of WS deployed on top of JXTA protocols. ManP2P management services are implemented as WS and can be accessed using the JXTA-SOAP libraries. Such services, in turn, are advertised in the ManP2P overlay using JXTA service advertisement messages. TLMs then learn the set of management services available on ManP2P either by actively querying the management overlay or by receiving management service advertisements.

As presented before, once a network device needs to issue a notification, the device does so by sending an SNMP trap message to a MLM. The network infrastructure required to transport such notification is a traditional TCP/IP network connecting the notifying device with at least one MLM. SNMP messages are run on top of UDP and uses IP routing to reach the MLM.

Once the MLM receives an SNMP trap, it checks its internal registries (controlled by the publish-subscribe mechanism presented before) to learn about the TLMs interested in the notification. For each TLM interested, the MLM sends the notification now using SOAP and the JXTA P2P infrastructure. In this case, the protocol stack additionally includes the SOAP envelope as well as the JXTA protocols. Routing is performed by peers at the JXTA application level. Figure 3 presents the life-cycle of a notification since its source device until it reaches the destination TLMs with the intermediation of MLMs.

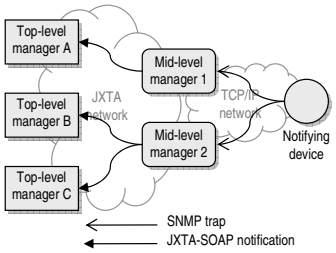


Fig. 3. Notification transport

In this example, a single network device notifies two MLMs. This notification can be realized by sending two copies of the same SNMP trap, each copy delivered to a different MLM using regular unicast TCP/IP communications. The problem with this approach is that with an increased number of MLMs a larger number of trap copies will be required, which consumes more bandwidth and processing power at the notifying device to create the additional copies. A more scalable alternative, however, is to use IP multicasting to issue a single trap that will still be delivered to all destination MLMs. This mechanism consumes less bandwidth and processing power but requires the underlying TCP/IP network to support IP multicasting, which is not always possible. In our experiments we have used this last option, placing the MLMs and notifying devices in the same IP network with multicast support enabled.

At the second communication stage we see the notification traveling from the MLMs towards the TLMs. Notice that in this case the MLM #1 is responsible for forwarding the notification to the TLM A, while the MLM #2 is responsible for forwarding the notification to the MLMs B and C. All associations between TLMs, MLMs, and managed devices are dynamically defined by auxiliary components omitted in this paper because the maintenance of these associations does not affect the evaluations to be presented. In a management environment similar to the one presented in Figure 3, we performed a set to experiments to understand the impact of WS-based notifications over P2P against SNMP traps, as will be presented in the next section.

IV. Evaluation

In this section used to perform the evaluation tests are presented showing the hardware and network setup. The results show the performance of P2P-based notification in terms of processing time at the MLMs and notification delivery delay from the source network device up to the TLMs. These results can guide the decision on the number of MLMs to be used according to a management system necessities.

A. Evaluation Scenarios

Our evaluations has been performed in a high-performance cluster where TLMs and MLMs has been deployed. The cluster is composed of 20 homogeneous nodes, each one composed of a Pentium III 1.1GHz processor, 512 MB cache, and 1 GB of RAM.

In order to evaluate the processing and forwarding delays at the MLMs, two evaluations scenarios have been used. In the first one, a single network device was responsible to notify TLMs issue SNMP traps forwarded by intermediate MLMs. The number of MLMs in this scenario varied from 1 up to 3, while the total number of TLMs varied from 1 up to 12. The target TLMs has been associated to the MLMs in balance way. For example, with 12 TLMs and 3 MLMs, each MLM was responsible to notify 4 TLMs.

In the second evaluation only 1 MLM and 1 TLM have been used, but the number of notifying network device has been increased, from 1 up to 12. The objective of this second scenario is to stress the intermediate single MLMs in order to observe its behavior. Figure 4 present these two evaluation scenarios.

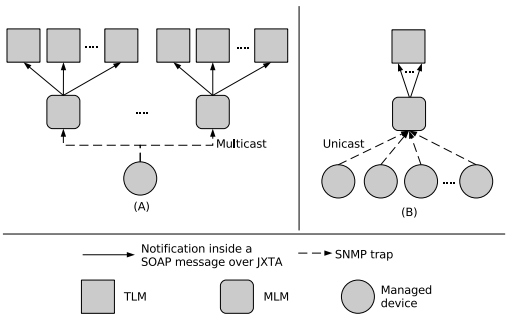


Fig. 4. Evaluation scenarios

B. Notification Propagation

In a previous work [19], as mentioned before, some results concerning the propagation delay of notifications in ManP2P has been presented. The main conclusion of those preliminary results was that the processing delay at MLMs was the most significant one, corresponding to 99% of the whole propagation delay of a notification, since its source device until the destination TLMs. The evaluations carried out in this present work allows to draw more precise analysis about the impact of MLMs over notification propagation.

The first results have been collected considering the management scenario where a single device needs to notify up to 12 TLMs using up to 3 MLMs, as presented before. For a proper balancing of TLMs among MLMs we strictly defined that each MLM in the same setup must be associated to the same number of MLMs. This however turns some setups impossible, for example, 4 TLMs notified by 3 MLMs: one of the MLMs will be responsible to notify 2 TLMs, leaving the other MLMs associated to a single TLMs. Table I presents the processing delay for each notification in the MLMs considering several setups. For those "impossible" setups an interpolation of values has been performed. Notice that the setup 1-TLM/2-MLMs, 1-TLM/3-MLMs, and 2-TLMs/3-MLMs are note presented because not all MLMs are required to notify all TLMs present.

TABLE I
MLMs average notification processing delay

TLMs	1 MLM	2 MLMs	3 MLMs
1	0,091146		
2	0,100734	0,092718	
3	0,110740	0,097300	0,093278
4	0,113827	0,101881	0,097023
5	0,116808	0,105282	0,100768
6	0,129409	0,108684	0,104513
7	0,136451	0,111527	0,106099
8	0,143494	0,114369	0,107684
9	0,167215	0,118814	0,109270
10	0,190936	0,123258	0,112102
11	0,202471	0,124930	0,114935
12	0,214006	0,126602	0,117767

The values from Table I has been used to create the graph presented in Figure 5, where the average processing delay is presented according to the total number of TLMs. The three lines correspond to the setups where 1, 2, and 3 MLMs have been used.

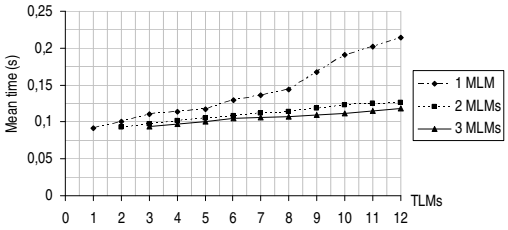


Fig. 5. Processing mean time

It is possible to observe that the processing delay in the setup with 1 MLM is sensibly greater than in the setups with 2 and 3 MLMs (these last two setups present very similar delays). In fact, regardless the TLMs considered, the processing delay is always greater using 1 MLM than using 2 or 3 MLMs. This values confirm the expected behavior,

where the introduction of additional MLMs to handle the same notification load (same number of target TLMs to be notified) leads to a decreased load in each MLM individually.

The single observation of the processing delay, although accurate, does not provides a clear view about performance improvements achieved with the inclusion of new MLMs. In order to have a better perspective on this issue, we check the *speedup*, which in this work represents the relative improvement on the notification performance, i.e., the relation between the time to sequentially execute a process and the time to execute the same process in parallel [20]. In our case, the process in question correspond to the activity of notifying a certain number of TLMs. Let t_1 be the time in the sequential execution (i.e., 1 single MLM forwarding all notifications), and t_N be the time in the parallel execution (i.e., N MLMs forwarding all notifications). The speedup $\sigma(N)$ of N MLMs is given by:

$$\sigma(N) = t_1/t_N \tag{1}$$

Given the previously values presented in Table I it is possible to compute two speedups: one for the setup with 2 MLMs and another for the setup with 3 MLMs. These speedups are calculated by dividing the column of 2-MLMs by the column 1-MLM (speedup 2), and by dividing the column of 3-MLMs by the column 1-MLM again (speedup 3). These speedups are presented in Figure 6. The setups using 1 and 2 TLMs have been suppressed because there were no values for 2 and 3 MLMs in these cases in Table I.

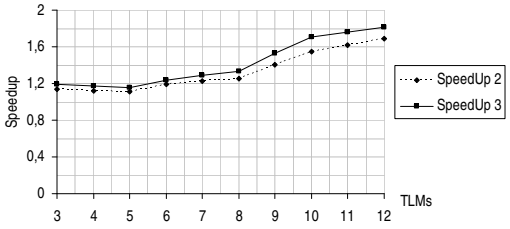


Fig. 6. Speedups of including new MLMs

We can take two main conclusions from Figure 6: the speedup of using 3 MLMs is greater than using 2 MLM, and both speedups increase when more TLMs need to be notified. A greater speedup for 3 MLMs than for 2 MLMs was expected: with more computing resources a better performance can be achieved. At the same time, however, the increasing speedups as a function of the number of TLMs should be surprising, since with more TLMs more notification load exists, which should supposedly decrease the total performance. The increasing speedup is explained in this case because the speedup indicates how efficiently the process is executed, i.e., with more TLMs more efficient become the MLMs employed. This is different than the processing delay presented in Figure 5, where the increased number of TLMs always leaded to an increased processing delay. In conclusion, the greater the number of TLMs to be notified the more efficient will be the additional MLMs employed.

This conclusion can in fact be further refined. Figure 6 shows that both speedups are in fact quite similar, which may leads one to reason whether the setup employing 3 MLMs would not be wasting computing resources. Why should one use 3 MLMs if the speedup for 2 MLMs is almost the same? In order to answer this question we propose the observation of an additional value: the relative speedup. This value takes into account not only the "regular" speedup considered before, but also the maximum speedup possible for a given setup, which corresponds to the number of MLMs employed. Thus, the relative speedup (σ_R) is defined according to the equation 2.

$$\sigma_R(N) = \sigma(N)/N \tag{2}$$

The relative speedups for 2 and 3 MLMs are presented in Figure 7.

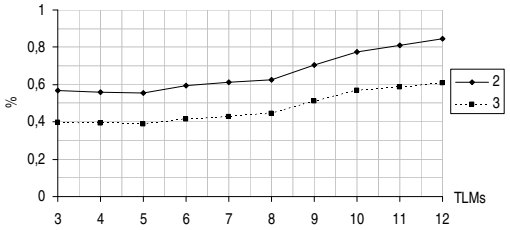


Fig. 7. Relative speedups

From Figure 7 it is possible to observe an inverted behavior: while Figure 6 the setups with 3 MLMs looked like the better option, in Figure 7 the opposite happens. Besides this inverted behavior, the lines become farther to each other and it is possible to observe that minimum values for $\sigma_R(2)$ correspond to the maximum values for $\sigma_R(3)$.

The explanation for this inversion is simple: speedup considers only the relationship between the sequential and parallel processing delays, but it does not consider the cost to achieve the reduced parallel processing delays. On the other hand, the relative speed (as defined in the equation 2) considers such a cost when it includes the maximum possible speedup.

C. Propagation Delay Variation

Every time a network device issues an SNMP trap, this trap can be sent to several MLMs, which in turn will forward the trap as a notification to various TLMs. In an ideal situation, all TLMs should receive the notifications at the same time allowing all TLMs to have the exactly same view of the managed devices. However, due to variations on the propagation delay, one TLM view may be different than another TLM view. In

order to evaluate the degree of variation in the propagation delay, additional experiments have been executed.

Figure 8 presents the minimum and maximum propagation delay considering the several setups previously presented.

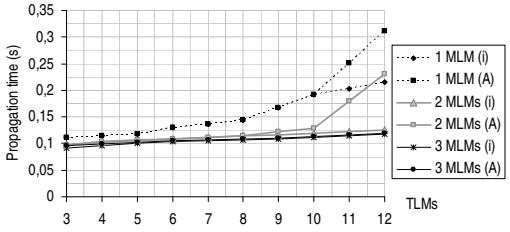


Fig. 8. Propagation delay variation

It is possible to observe from Figure 8 that, for a small number of TLMs, there is almost no variation in the propagation delay, but once the number of TLMs increases the variation increases as well. The degree of increase when using 3 MLMs does not exist. The relative difference between the maximum and minimum in each setup are listed in Figure 9.

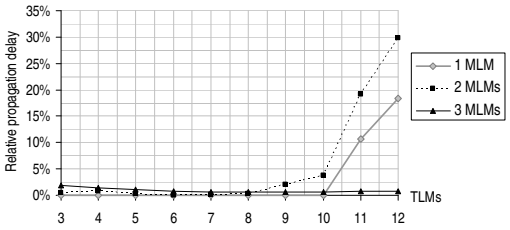


Fig. 9. Relative difference of propagation delay

Again, for fewer TLMs, almost no variation exist, but after 8 or more TLMs the variation increases, except when using 3 MLMs.

Another aspect that needs to be analyzed is the effect of a trap overload at the MLMs, mainly when we remember that the MLM processing delay dominates the propagation delay of a notification. To proceed with this another analysis, the second management scenario, where an increasing number of devices will notify a single TLM using a single intermediate MLM, will be used.

The values measured for this second scenario are presented in Table II. Columns minimum, maximum, and average present values for 2, 4, 8, and 16 notifying SNMP-enabled devices.

The values from Table II are depicted in Figure 10 that shows that the relationship between the average propagation delay and the number of notifying devices are linear.

Using the values to depict the Figure 10, the propagation delay can be estimated according to the equation:

$$t_P(N) = 0,097535N - 0,03402 \quad (3)$$

TABLE II
NOTIFICATION PROPAGATION DELAY FOR A TRAP OVERLOAD

Devices	Minimum	Maximum	Average
2	0,175231	0,175232	0,175232
4	0,350290	0,350300	0,350295
8	0,730155	0,730170	0,730161
16	1,534265	1,53429	1,534279

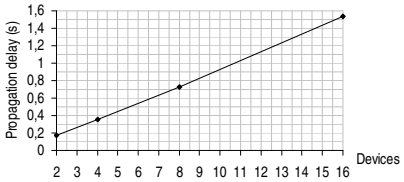


Fig. 10. Graph of the notification propagation delay for a trap overload

where N is the number of notifying devices and $t_P(N)$ is the average propagation delay of a trap from the notifying device up to the TLM.

The analysis of the variation in the propagation delay for the second scenario will be similar to that used in the first scenario. Observing the values previously presented in Table II it is possible to observe that the minimum and maximum propagation delays are quite similar regardless the number of notifying devices, which indicates a very deterministic behavior.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have evaluated the use of WS and P2P technologies to transport notifications in a management overlay that integrates SNMP-enabled devices in the system architecture. Through the ManP2P solution, a set of experimental evaluations has been carried out in order to understand the impact of the P2P overlay over the notification performance.

First we conclude that with the introduction of additional MLMs in the management overlay the load of notifying a set o TLMs is properly decreased. In order to understand the scalability of the notification support, two values need to be considered: the processing delay at each MLM, and the relative speedup. The processing delay presented in this paper leads us to conclude that the greater the number of MLMs the less the propagation delay. On the other hand, the relative speedup indicates that the cost of including new MLMs increases with the number of MLMs, which limits that total amount of MLMs to be used. Choosing a specific setup to manage a production environment should then take these two value into account.

The analysis of the factors that could lead to increased variations in the notification propagation delay shows that, for the setups used in our evaluations, the variations stayed confined inside acceptable values. Even in the case with up to 3 MLMs notifying up to 12 TLMs, where the relative difference reach 30%, it is possible to notice that the absolute values are small, around 0.2 seconds, which probably will not affect the human perception on the notification arrival at the TLM side.

For future work we plan to execute additional evaluations considering a network communications infrastructure more hostile than the one provided by the high-performance cluster used in this paper. For example, we will observe the behavior of notifications if the Internet is the mean to connect MLMs and TLMs. In addition, we will execute tests with a larger number of TLMs executing in the global Internet. These next evaluations are planned to be performed using the PlanetLab infrastructure.

REFERENCES

[1] D. Harrington, R. Presuhn, and B. Wijnen, "RFC 3411 - An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," December 2002, available at: http://www.ietf.org/rfc/rfc3411.txt.

[2] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web Services Web - An Introdution do SOAP, WSDL,and UDDI," *IEEE Internet Computing*, vol. 6, no. 2, pp. 86–93, 2002.

[3] A. Pras, T. Drevers, R. v.d. Meent, and D. Quartel, "Comparing the Performance of SNMP and Web Services-Based Management," *IEEE eTNSM - eTransactions on Network and Service Management*, vol. 1, no. 2, p. 11, Dec. 2004.

[4] W. Vambenepe, "Web Services Distributed Management (WSDM): Management Using Web Services (MUWS 1.0) Part 1," 2005, http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part1-1.0.pdf.

[5] A. Arora, J. Cohen, J. Davis, E. Golovinsky, J. He, D. Hines, R. McCollum, M. Milenkovic, P. Montgomery, J. Schlimmer, E. Suen, and V. Tewari, "Web Service for Management (WS-Management)," http://developers.sun.com/techtopics/webservices/management/WS-Management.June.2005.pdf, Jun. 2005.

[6] R. State and O. Festor, "A Management Platform Over Peer-to-Peer Service Infrastructure," in *Proceeding. 10th International Conference on Telecommunications, 2003. ICT 2003*, Vancouver, BC, Canada, 2003, pp. 124–131.

[7] L. Z. Granville, D. M. da Rosa, A. Panisson, C. Melchiors, M. J. B. Almeida, and L. M. R. Tarouco, "Managing Computer Networks Using Peer-to-Peer Technologies," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 62–68, 2005.

[8] N. Mitra, "SOAP Version 1.2 Part 0: Primer," June 2003, available at http://www.w3.org/TR/2003/REC-soap12-part0-20030624/.

[9] N. Catania, P. Kuma, B. Murray, H. Pourhedari, W. Vambenepe, and K. Wurster, "Web Services Events (WS-Events) Version 2.0," http://devresource.hp.com/drc/specifications/wsmf/WS-Events.pdf, Jul. 2003.

[10] P. Niblett and S. Graham, "Events and Service-Oriented Architecture: The OASIS Web Services Notification Specifications," *IBM Systems Journal*, vol. 44, no. 4, pp. 869–886, 2005.

[11] L. Gong, "JXTA: A Network Programming Environment," *IEEE Communications Magazine*, vol. 5, no. 3, pp. 88–95, May 2005.

[12] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies," *ACM Comput. Surv.*, vol. 36, no. 4, pp. 335–371, 2004.

[13] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," Internet Engineering Task Force (IETF), RFC 3411, STD 62, Dec. 2002.

[14] A. Binzenhfer, K. Tutschku, B. auf dem Grabem, M. Fiedler, and P. Carlsson, "A P2P-Based Framework for Distributed Network Management," in *Proceedings. Wireless Systems and Network Architectures in Next Generation Internet*, ser. Lecture Notes in Computer Science, vol. 3883. Heidelberg, Springer-Berlim, 2006, pp. 198–210.

[15] M. Brunner, A. Galis, L. Cheng, J. A. Cols, B. Ahlgren, A. Gunnar, H. Abrahamsson, R. Szabo, S. Csaba, J. Nielsen, A. G. Prieto, R. Stadler, and G. Molnar, "Towards Ambient Networks Management," in *Proceedings. Mobility Aware Technologies and Applications*, ser. Lecture Notes in Computer Science, vol. 3744. Heidelberg, Springer-Berlim, 2005, pp. 215–229.

[16] J. Schonwalder, J. Quittek, and C. Kappler, "Building distributed management applications with the IETF Script MIB," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 5, pp. 702–714, 2000.

[17] P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, June 2003.

[18] M. Amoretti, F. Zanichelli, and G. Conte, "SP2A: A Service-Oriented Framework for P2P-Based Grids," in *MGC '05: Proceedings of the 3rd international workshop on Middleware for grid computing*. New York, NY, USA: ACM Press, 2005, pp. 1–6.

[19] C. R. P. dos Santos, C. C. Marquezan, E. M. Salvador, . L. F. D. Santa, S. L. Cechin, and L. Z. Granville, "On the design and performance evaluation of notification support for p2p-based network management," in *ISCC 2007: 12th IEEE Symposium on Computers and Communications (submitted)*. IEEE Computer Society, 2007.

[20] R. Buyya, Ed., *High Performance Cluster Computing: Programming and Applications*, 1999.

An Evaluation of Service Composition Technologies Applied to Network Management

Ricardo Lemos Vianna, Everton Rafael Polina, Clarissa Cassales Marquezan, Leandro Bertholdo,
Liane Margarida Rockenbach Tarouco, Maria Janilce Bosquiroli Almeida, Lisandro Zambenedetti Granville
Institute of Informatics – Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 - Porto Alegre, RS - Brazil
Email: {rvianna, polina, clarissa, berthold, liane, janilce, granville}@inf.ufrgs.br

***Abstract*— Service composition is a technique that may help the development of management systems by aggregating smaller services to produce more sophisticated ones. Service composition can be realized by using traditional management technologies, although these technologies have not been conceived taking composition support as one of their main aspects. Current service-oriented architecture (SOA)-related efforts, however, define specific standards for Web services composition, such as the Web Services Business Process Execution Language (WS-BPEL). Web services for network management have been investigated by the management community at least in the last four years, but up to today no research evaluating Web services composition applied to network management has been carried out. In this paper we present such an evaluation where compositions based on the IETF Script MIB, *ad-hoc* Java Web services, and WS-BPEL are compared against one another in a managed network where BGP routers are investigated in order to identify route advertisement anomalies.**

I. Introduction

Service composition [1] is a technique used to aggregate or combine services in order to build up new, more sophisticated ones. It is also a core element of the service-oriented architecture (SOA) [2], which in its turn is the key architecture for the modern Web-based systems. As a technique, service composition can be used to address problems of several computer science disciplines, including network management, where composition is especially interesting when a complex management process requires the execution of smaller activities in order to be successfully accomplished. For example, to track the number of routes an autonomous systems (AS) advertises through its different routers, a composition that combines the routers' information exposed by their management agents is required, so one can be able to detect, for example, possible anomalies on an AS behavior.

Service composition itself is not new in computer science, but the efforts towards the definition of standards for service composition have initiated only around the last five years. With the lack of proper standards, service composition in network management has been manually realized using traditional management technologies, but not without heavy coding efforts usually combined with low flexibility. This is so because network management technologies have no "native" support for service composition on their core components, forcing composition to be implemented via particular solutions.

The current researches and standards for service composition are mainly focused on coordinating the interactions among Web services deployed along the Internet [3] [4] [5]. One of these standards - WS-BPEL (Web Services Business Process Execution Language) [6] - is strongly based on the workflow approach to provide properly orchestrated communications of Web services participating in a composition. One of the most important aspects about such standards, and particularly about WS-BPEL, is that they allow the definition of compositions in an easier and more proper way when compared with the *ad-hoc* compositions that have been carried out so far in network management.

This ease of use, however, is achieved with the price of increased processing delays and additional network bandwidth consumption, due to extensive exchange of XML-based messages. Considering the network management field, Web services-based management is not a new research area, but up to today there is no investigation that has determined whether and how the service composition standards could improve the composition of management services, replacing the composition solutions normally used in network management. We believe that Web services composition can really bring interesting opportunities for network management, but at the same time, possible drawbacks can prevent its use. The main contribution of this paper thus relays on the evaluation of service composition solutions for network management which we have carried out in order to clarify and understand the pros and cons of employing Web services composition for network management.

The remainder of this paper is organized as follows. In Section 2 a review of service composition in the context of network management is presented. Additionally, in Section 2 we also briefly introduce the WS-BPEL standard. Our evaluation has been carried out considering a country-wide backbone where BGP routers need to be investigated to detect route advertisement anomalies. This management environment and the target composition associated to it are presented in Section 3. The investigated service composition has been modeled and implemented considering three different approaches: compositions based on the IETF Script MIB, *ad-hoc* compositions of Web services management gateways, and compositions described in WS-BPEL documents. These composition approaches and their respective implementations

are discussed in Section 4. In Section 5 we present a set of evaluating tests executed over the management environment of BGP routers. Tests and results are analyzed and discussed in order to draw the main conclusions of this paper, which is finally closed in Section 6, where final remarks and future work are presented.

II. Background

With the fast deployment of new services in networked environments, several management activities are initially manually performed by network administrators while no automation for such activities is supported in management software. The complexity of management activities may vary from simple queries directed to managed devices to complex calculations using information retrieved from different remote locations. In this last case, service composition can represent an interesting tool to build up more sophisticated services based on the combination of less complex ones.

As mentioned in the introduction section, service composition itself is not a new technique, and in fact can be realized using traditional management technologies. However, we believe that the employment of technologies specifically created to support service composition could bring important advantages to the network management discipline. In this section we first review how service composition can be implemented using traditional management technologies. After, we briefly introduce *ad-hoc* compositions to then close the section with the presentation of the WS-BPEL standard used in our evaluations.

A. SNMP and Service Composition

Probably, the most frequent service composition in network management occurs when network administrators code their personal bash scripts to perform an activity composed of smaller actions. Often, however, the results of the execution of such a composition are confined to the execution environment, and no other external software can use them to build up new compositions. We consider that proper compositions are characterized not only by the agglutination of smaller services to form a more complex one, but also by the ability of the composed service to expose its results to serve as the basis for the definition of additional and even more sophisticated services in a chain or hierarchy of compositions. In this sense, compositions made coding bash scripts cannot be considered proper compositions.

A more adequate option for service composition in network management is the use of the Simple Network Management Protocol (SNMP) [7] as a mechanism to expose the composed services. For example, RMON [8] and RMON2 [9] MIB objects expose compositions of management information collected by management probes located on dedicated devices or internal to routers and switches. In this case, SNMP is used only to expose the composed information, since the original information is retrieved not using SNMP but sniffing the network segments of interest.

In an all-SNMP composition solution, however, SNMP compositive agents can be coded to contact remote agents and combine the information retrieved from them. The results of such processing (i.e., the results of the composition) are then exposed to other higher-level agents also via SNMP. Astrolabe [10] and the work developed by Praveen Yalagandula and Mike Dahlin [11], for example, use SNMP-based compositions to build hierarchical levels of management information, where information from the leafs of the system are composed to express the whole status of the managed network. This approach resembles the management by delegation (MbD) model [12], where intermediate entities in a management hierarchy are dual-role: they are managers when accessing lower-level agents, and agents when exposing information for higher-level managers. These intermediate entities are usually referenced as mid-level managers in the management literature. Figure 1 depicts a set of cascading mid-level managers used to compose management services.

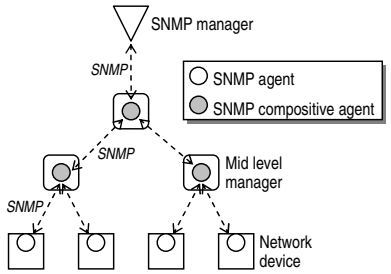


Fig. 1. Compositions coded in SNMP agents

A key problem of the previous approach is that if the composition needs to be changed for some reason (e.g., a different calculation is required in the mid-level managers), the SNMP agents need to be recompiled, which is usually expensive. A more flexible SNMP-based approach is the use of the IETF Script MIB [13]. In this case, SNMP is used as a script transfer and execution control mechanism. A network manager initially transfers via SNMP a script to mid-level managers that execute the management script using an internal runtime engine, such as a Java virtual machine or a TCL interpreter. Figure 2 shows the general approach when using the Script MIB.

It is important to notice that the composition logic in the Script MIB solution is now coded on the management scripts, instead of internally to the SNMP agents of the mid-level managers. Thus, the installation of new compositions requires only the transfer of new scripts, having no necessity of recompiling the SNMP agents anymore. Another important point is the fact that the selection of the language used to define the compositions depends on the execution environments available on the remote managers. Additionally, in order to have a hierarchy of service composition, the language used needs to have support for SNMP because when a script in

execution needs to contact a remote SNMP entity it does so using the language's SNMP support. Considering this, building compositions in a large hierarchy with several levels of mid-level managers is not an easy task because, as mentioned before, SNMP-based technologies, including the Script MIB, have not been defined with composition in mind.

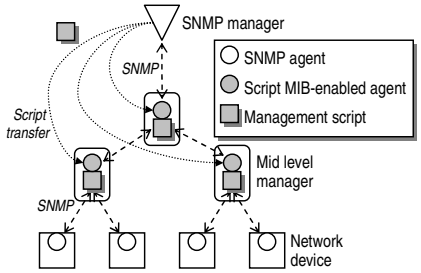


Fig. 2. Compositions based on the Script MIB

Recently, the IETF has been working on the definition of the XML-based NETCONF [14] protocol, devoted to network configuration. Although the employment of NETCONF would form a more elegant composition solution when combined with WS-BPEL (to be presented ahead), very few (if any) actual network devices support NETCONF. Since we are focused in evaluating the composition considering real scenarios, we won't address NETCONF in the remainder of this paper, even though we are aware of the NETCONF importance in the network management field.

B. Ad-hoc Compositions

We use the term "*ad-hoc* compositions" to address compositions manually coded using interpreted scripts or programming languages, and being based on no specific solution originally defined to support service compositions. In this paper, we also assume that *ad-hoc* compositions use SOAP (Simple Object Access Protocol) [15] as the protocol to communicate the compositive software with the basic services to be composed. An example of an *ah-hoc* composition is Web meta-search engines, i.e., engines that contact other ones to search for information and aggregate the results to provide a unified view of them to the user that requested the original search. Book search engines are an example of popular meta-search engines on the Web. In these systems, the communication between the meta-search engine and the third-party engines is based on SOAP, but probably not specified using a composition standard such as WS-BPEL.

An example of an *ad-hoc* composition in network management is presented in the XMLNET management system [16]. XMLNET is extensively based on XML, which used as the basic representation of management information. In order to integrate SNMP-enabled devices in the management system, XMLNET uses SNMP to XML gateways. The communication among the Web components of the system is performed using XML-RPC [17] instead of SOAP. XMLNET is developed in Java, and the service compositions are based on a non-standardized language defined by the system authors.

The advantage of the *ad-hoc* composition approach over the SNMP-based approaches presented before is that the use of SOAP as the communication mechanism is usually more appropriate for communications over the Internet. In addition, even for retrieving management information from network devices, SOAP performs better than SNMP if a large number of management variables is exchanged [18]. Although SNMP devices will not be replaced in a short-term by Web services-enabled devices, SNMP to SOAP gateways can effectively integrate SNMP devices in Web services-based management systems [19], thus allowing "legacy" devices to participate in a composition hierarchy using SOAP. Figure 3 presents a sample environment where SNMP devices are integrated in a Web services-based compositive system.

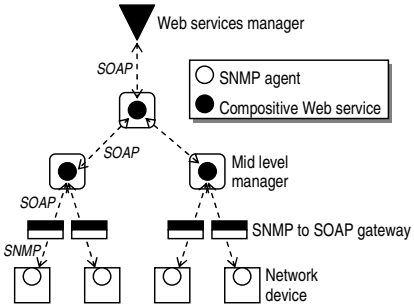


Fig. 3. Web services *ad-hoc* and WS-BPEL compositions

Although more interesting than the SNMP approaches for composition, the *ad-hoc* composition still presents the lack of flexibility usually required in dynamic management environments. If the composition needs to be somehow changed, the composition code needs to be rewritten (and recompiled if a programming language is used rather than a scripting language). This limits the applicability of service composition in more dynamic environment, and that in fact has motivated the development of composition-specific standards, such as WS-BPEL.

C. Web Service Business Process Execution Language

WS-BPEL (Web Services Business Process Execution Language) [6] is probably the most relevant and well accepted standard for Web services composition. Until version 1.1, WS-BPEL specification was called BPEL4WS (Business Process Execution Language for Web Services). When the working draft that proposed a new version (2.0) was released, it also changed the specification name to WS-BPEL. The standardization of WS-BPEL is currently under OASIS's arms, which released the newest specification draft in May, 2006. Our evaluation tests are based on BPEL4WS because the composition engine used (ActiveBPEL [20]) has not been

updated by its developers to support WS-BPEL yet. Although we use BPEL4WS, WS-BPEL and BPEL4WS share the same main principles.

WS-BPEL models the behavior of a composition through an XML grammar that describes the logic needed to coordinate the services that participate in a process flow. That grammar is interpreted and its stated actions executed by a composition engine, such as ActiveBPEL, that coordinates the activities using a compensation strategy when errors occur.

Basically, WS-BPEL is a new layer built on the WSDL standard, where WSDL define operations, partners, and data types involved in the composition and WS-BPEL establishes how those operations will be sequenced. WS-BPEL supports basic and structured activities. Basic activities can be seen as a component that interacts with things externals to the own process, such as manipulating requests and replies or invoking external Web services. Structured activities, on the other hand, manage the entire process flow, specifying, for example, if some tasks should run sequentially or concurrently.

Using a composition standard one can compose services to create new services as if one was just modeling a workflow[1], in a higher level if compared to *ad-hoc* compositions. In WS-BPEL compositions, the user only needs to care about the logic of the new composed service, instead of worrying about the logic and how to implement it using a particular programming language and API (as in the *ah-doc* approach). In other words, implementation details are hidden from the user by using a composition standard. Other advantage brought by such standards is that they inherit all the advances resulted from previous workflow researches (e.g., formal semantic issues) since workflows and service compositions are very similar. Fault tolerance aspects are also covered by WS-BPEL standard. It has powerful mechanisms, such as roll back when some point of the composition fails, which allows one to create "transactional" services. The previously presented Figure 3 illustrates a WS-BPEL Web services composition for network management as well. Despite the different implementations, *ad-hoc* and standardized compositions share the same general architecture.

III. Management Environment

In order to evaluate the composition solution presented before, we have coded a set of compositions intended to manage some Brazilian internet exchange points (Brazilian IXPs - PPT-Metro project[2]) and their relationship with several autonomous system (AS) peers, in special with the country-wide Brazilian National Education and Research Network (RNP)[3]. Remote autonomous systems (ASes) connected to RNP constantly advertise BGP routes in 12 Internet exchange points (IXPs) located along the 27 RNP's points of presence (POPs). This environment needs to be managed because the same remote AS may advertise different routes in different IXPs, which leads to routing anomalies or peer-agreement violations.

Through the composition of routing and connectivity information obtained from IXPs, and using IPXs different routing views information it is possible to make several inferences about national Internet stability and growth. Moreover, such compositions can indicate regions in growth, if considered the increase of prefixes announced throughout the years in each IXP. Thus, based on the number of new routes advertised, it is possible to measure if the economy of a certain region is increasing or decreasing, and drive the government investments on the Internet initiative. Based on this kind of information it is also possible to establish quality levels for the Internet in each part of the country. For example, based on these established levels it is possible to firm SLAs with partners that will have to adjust their ASes to the quality level on that region.

By checking some BGP parameters and drawn prefixed on a IXP it is possible to measure regions that are suffering some disruption, like a dissemination of a computer virus, like those occurred in 2001 (CodeRedv2) and 2003 (W32.Slammer) that has shut down several minor ISPs around the planet, impacting on the global and national BGP table; that approach can measure the impact on Internet when accident or vandalism involving optical fiber disruption and another infrastructure problems happen. In essence, BGP-related information collected in the country-wide backbone drives the governmental investments on the national Internet initiative. Without the knowledge about the BGP advertisements, the investments may be guided towards wrong directions.

The management of BGP routes has been already addressed in the past. For example, Musunuri and Cobb [21] have investigated the divergences on AS tables and presented a survey listing possible solutions. Dimitropoulos and Riley [22], in turn, have presented an investigation on modeling AS relationships by simulating the Internet topology. We focus here on the necessity of monitoring remote ASes through different IXPs in order to detect possible anomalies. That is accomplished by management service composition.

In our solution, service composition for the management of the RNP's BGP border routers happens in two contexts. First, the composition of management information found in a single router is required to compute the number of routes a specific AS has advertised to a specific gateway. Another level of composition happens when information from different routers need to be aggregated to calculate the overall advertisement activity an AS is posing in the whole RNP backbone. Figure 4 depicts the managed environment highlighting the composition contexts.

Each BGP router may connect different ASes. Each AS, in its turn, may be connected to the RNP backbone through different BGP routers in different IXPs. A top-level manager is responsible for monitoring the advertisement patterns of each remote AS connected to RNP possibly via multiple IXPs. To do that, the top-level manager acts as a BGP monitor that contacts the mid-level manager of level 1 requesting a table of advertisement information for a giving AS. For example,

[1]In fact, there are graphical tools that aid users to create workflows and generate associated WS-BPEL documents

[2]http://www.ptt.br

[3]http://www.rnp.br

considering the network shown in Figure 4, the request of the advertisement table of AS number 3 would result in the Table I.

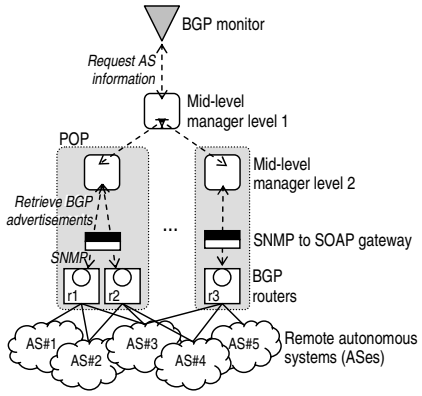


Fig. 4. RNP country-wide backbone with service composition

The difference in the number of advertisements sent to each BGP router may indicate, as mentioned before, routing anomalies that should be addressed. In order to produce the Table I output, the mid-level manager of level 1 composes the management information retrieved from mid-level managers of level 2 in each RNP POP that host one of the 12 IXPs. Mid-level managers of level 2, in turn, retrieve management information from the POP local routers accessing, via SNMP, the IETF BGP4 MIB [23]. If the composition implemented by the level 2 mid-level manager is based on Web services, then an intermediate SNMP to SOAP gateway is placed between the mid-level manager and the target device. If this is the case, POPs with more than one border BGP gateway share a single gateway to convert SNMP to SOAP messages. If the composition solution is solely based on SNMP, no gateway is required.

TABLE I
ADVERTISEMENT TABLE FOR AS NUMBER 3

AS♯3 advertisement table	
BGP router	Number of advertisements
Router r1	181
Router r2	663
Router r3	36

In the following section we describe three implementations used to investigate SNMP and Web services-based technologies for service composition applied to the management of the RNP's BGP routers advertisements.

IV. IMPLEMENTATION

In order to support the management of the previously presented environment, we have coded three solutions based, each one, on the Script MIB, on *ad-hoc* composition, and on WS-BPEL, respectively. All compositions perform operations in two levels, using the support of mid-level managers of levels 1 and 2, as shown in Figure 4.

At the bottom of the architecture, each level 2 mid-level manager contacts local BGP routers agents to retrieve the advertisements of an AS of interest. That is always performed via SNMP, since BGP routers do not natively support Web services-based management interfaces. Two objects of the BGP4 MIB are of special interest here: `bgpPeerRemoteAs` and `bgp4PathAttrPeer`. With proper treatment of these objects one can retrieve the list of advertisements associated to an AS.

The retrieval and manipulation of the values associated with `bgpPeerRemoteAs` and `bgp4PathAttrPeer` are performed by the level 2 mid-level manager. The result of this manipulation is a single pair listing the BGP router and associated number of advertisements. This composed information is exposed to the level 1 mid-level manager that performs the second service composition. The communication between level 1 and level 2 mid-level managers depends now on the composition solution. If Script MIB is used, SNMP is the communication mechanism employed. For *ad-hoc* and WS-BPEL compositions, SOAP is used instead.

Further details specific to each composition solution are presented in the following sub-sections.

A. Script MIB Composition Details

In order to support compositions based on the Script MIB, the managed environment needs to provide Script MIB-compliant agents at the mid-level managers, as well as an execution engine to run the compositive scripts. To implement the mid-level managers we have used Jasmin [24], which is an implementation of the Script MIB developed by the Technical University of Braunschweig and NEC C&C Research Laboratories. Jasmin implements the Script MIB published in the RFC 2592, which was later updated by the RFC 3165.

Jasmin supports both Java and TCL runtime engines, so that the top-level manager and the level 1 mid-level manager can delegate Java and TCL management scripts to the Jasmin-based mid-level managers. Our compositions have been coded in two Java scripts: one of them to be executed in the level 1 mid-level manager, and the second one to be placed in the level 2 mid-level manager.

Mid-level managers' software infrastructure is composed of Jasmin version 1.0.0, Java Development Kid 118, SNMP support provided by the ucd-snmp package version 4.2.6, and Linux Suse distribution 6.4 (2.2.14). Although newer versions of these softwares are available, the Jasmin software, which is not maintained by their developers anymore, imposes some restrictions on the versions of the other software packages used.

The compositive script at the level 1 mid-level manager requests the script on the level 2 manager to be executed setting the `smLaunchStart` Script MIB object. Than the level 1 mid-level manager loops consulting the level 2 mid-level manager waiting for the end of the script execution.

That is done checking the `smRunState` object. When its state evolves to `terminated`, the level 1 mid-level manager retrieves the result of the composition on the level 2 mid-level manager accessing the `smRunResult` object. In fact an SNMP trap message is issued to indicate that the execution of a management script is over. However, since SNMP traps are UDP messages not acknowledge by the receiving manager and UDP messages may get lost more easily in hostile network environments such as the one where our system is intended to run, the safer way to ensure that a script execution is over is by polling the remote mid-level manager.

B. Ad-hoc Composition Details

Ad-hoc compositions have also been coded in Java, but instead of being transferred to a Script MIB-based mid-level manager, they have been statically installed as a regular Java software. Since no special transfer mechanism is required, the *ad-hoc* composition software infrastructure is not limited by the previous Jasmin package requirements. On the other hand, since in the *ad-hoc* composition the communications are based on SOAP, proper SOAP support needs to be provided.

In order to build up the software infrastructure of a mid-level manager to support *ad-hoc* compositions, the following software has been installed: net-snmp version 5.1.1, J2SDK 1.4.2, Apache Tomcat 5.0.28, and Apache Axis 1.2RC2.

Since the final BGP router exposes the management information through the SNMP BGP4 MIB, an intermediate SNMP to SOAP gateway has been used. Such gateway has been automatically generated using a gateway creation tool [25] that we have developed for previous Web services for management investigations. The gateway presents Web services operations for each BGP4 MIB object, allowing a higher-level manager to access the BGP4 information by invoking such operations.

Although a gateway has been introduced, it has been physically placed on the same host that runs level 2 mid-level managers. When such manager wants to retrieve BGP information from an SNMP managed device it first contacts the local gateway via an internal SOAP call to the gateway, which then forwards the request now using SNMP. Figure 5 shows the physical placement of each manager in an *ad-hoc* composition setup.

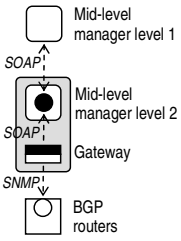


Fig. 5. Physical placement of mid-level managers and gateway

C. WS-BPEL Composition Details

As mentioned before, we have used ActiveBPEL to implement the composition support at mid-level managers. To install ActiveBPEL, the same software infrastructure used in the ad-hoc compositions has been used, with the additional installation of the AtiveBPEL itself, version 1.1.

The WS-BPEL compositions running on both level 1 and 2 mid-level managers have been specified with the use of a software tool named Network Information Aggregator (NIA) [26]. NIA helps network operator to define management services by using as information descriptor the traditional SNMP MIB modules.

The actual retrieval of management information from within the final BGP routers, again, requires the SNMP to SOAP gateways mentioned before.

V. Evaluation

The tests performed in this work aim at determining the response time and bandwidth consumption for each service composition implementation. To execute these tests we have deployed the mid-level managers presented before on the managed network and proceed with the measurements. Our experiments were carried out in a lab environment composed by two computers, connected via an 100Mbps switch, whose hardware setups for the top-level and mid-level managers are presented in Table II. It is important to mention that we intended to evaluate the possible solutions without the introduction of optimization. This question will be more apparent ahead.

TABLE II
Testing host description

	Top-level manager	Mid-level manager
Processor	AMD Athlon 2GHZ	AMD Athlon 2GHz
Cache	256KB	256KB
Memory	1GB	235MB
Swap	500 MB	800MB

Two different service composition levels has been observed in our evaluation, i.e., *device composition* and *network composition*. In the device composition, just one BGP router is contacted, and the number of advertisements of such router varies from 10 to 130. In network composition different services are contacted to form the more sophisticated one. In this case, the level 1 mid-level manager contacts all level 2 mid-level managers to build up an advertisement table given a specific AS of interest. In this last case we fixed the number of advertisements of each router to 10, but vary the number of mid-level managers from 1 to 10.

Figure 6 shows the environment setup to measured network usage and response time for the device composition.

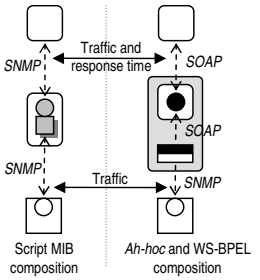


Fig. 6. Tests setup for device composition measurements

The environment for the measurements for the network composition is complementary depicted in Figure 7.

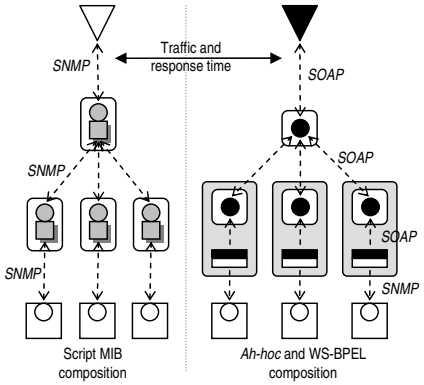


Fig. 7. Network composition tests

In the following subsections we present the results of this evaluation.

A. Network Usage

Network usage was measured in two different points for each service composition implementation. Regarding the Script MIB implementation, it was measured between the top-level manager and the mid-level manager, and between the mid-level manager and the final device's SNMP agent. Network usage for the *ad-hoc* and WS-BPEL service compositions has been measured between the Web services manager and the composition (mid-level manager), and between the SNMP to SOAP gateway and the SNMP agent (Figure 6).

Specifically on the Script MIB evaluation, the network usage includes the traffic introduced by the preparation of the script for execution, the traffic for monitoring the agent to detect the end of the script execution (using a polling operation of 10ms of interval), and the traffic generated to retrieve the execution results. The polling interval is 10ms because a previous software used by the RNP operators polled the remote entities using this interval. The time spent to transfer the script to the Script MIB agent was not computed because we considered that the script has been already deployed in the Jasmin agent. Furthermore, our measurements include all overhead from the lower layer protocols, i.e., transport, network, and data link layers. Figure 8 shows the network usage when retrieving 10 to 130 routes from a BGP router.

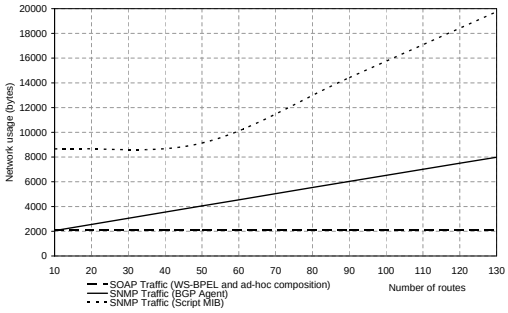


Fig. 8. Network usage for device composition

The solid line shows the SNMP traffic generated to retrieve the routing information directly at the BGP routers. Above the level 2 mid-level manager we have additionally measured the network usage imposed by the Script MIB, the *ad-hoc* composition, and the WS-BPEL composition. As one can observe, the Script MIB composition consumes more bandwidth due to the polling mechanism used by the Script MIB client to detect when the execution of the compositive script has finished. Both *ad-hoc* and WS-BPEL compositions, in turn, consumed a constant bandwidth regardless of the number of routes advertised. This is so because the composition executed in the level 2 mid-level manager reduces all routes retrieved from the BGP routers to a pair (router, number of advertisements), which consumes a fixed number of bytes to be transferred to the mid-level manager.

This network usage pattern is similar to the one previously published by Fioreze *et al.* [19]. Since the level 2 mid-level manager composes all advertised routes to produce a single pair of information, all SNMP traffic is confined to the mid-level manager and agent segment. The exception of this occurs when the Script MIB is used. Since there is no way for a Script MIB client (in this case, the level 1 mid-level manager) to safely learn that a script execution has finished except by polling the Script MIB agent, the bandwidth consumption will be increased.

Figure 9 presents the network usage considering the network composition. In this case, the traffic observed is that one between the top-level manager and the level 1 mid-level manager. Again, due to the polling mechanism used to access the Script MIB, the traffic generated in this composition is greater than in the other options. The traffic from the *ad-hoc* and WS-BPEL compositions is again constant.

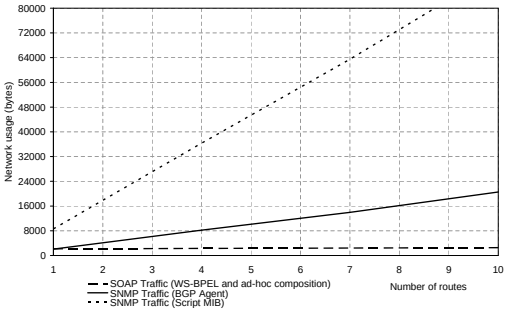


Fig. 9. Network usage for network composition

From these results we can conclude that, independently of using ad-hoc or WS-BPEL compositions, this Web services-based options are better than the solution based on the Script MIB. It is important to highlight, however, that this is most a consequence of the inability of the Script MIB of safely notify its clients about the end of a script execution.

B. Response Time

The response time is the time difference between the first message requesting an operation and the last message with the response associated to the request. For the device composition, the response time is observed between the level 1 and 2 mid-level managers. In this case, this level 1 manager requests to a single level 2 manager the number of advertisements a specific AS has issued in a BGP router. Internally, the level 2 manager sends several requests to the BGP router agents until the desired information is ready to be sent back to the level 1 mid-level manager. Figure 10 presents the response time associated to device compositions. In order to guarantee the statistic validation of the results, the experiments have been performed considering a confidence interval of 95% and running over 30 interactions.

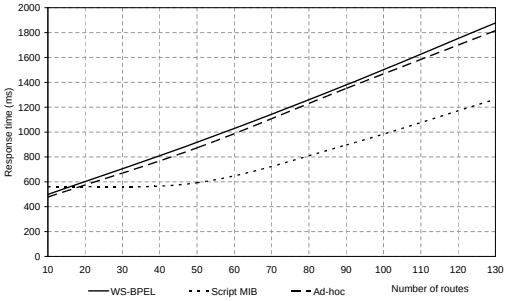


Fig. 10. Device composition response time

It is interesting to notice that the Script MIB composition often presents lower response time, although it also consumes more bandwidth as observed before. The low response time is due to low processing power required to process SNMP messages, which contrasts with the *ad-hoc* and WS-BPEL solutions, where more processing power is needed to handle all verbose, XML documents that form the SOAP messages.

Another interesting point comes from the difference between the response time associated to the *ah-hoc* and WS-BEPL compositions. Although WS-BPEL requires an additional execution engine to operate (in the case of our investigation the execution engine is provided by ActiveBPEL), no significant increase in the response time is observed when compared with the *ad-hoc* composition.

Figure 11 presents the response time now considering network compositions. Again, this response time has been calculated observing the communications between the top-level manager and the level 1 mid-level manager.

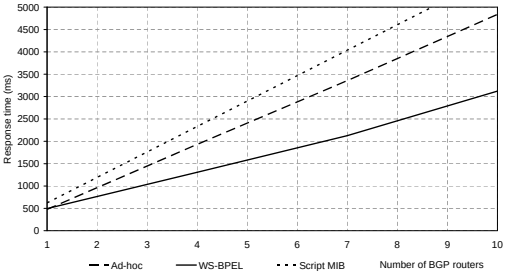


Fig. 11. Network composition response time

In the network response time, the Script MIB has presented worse performance than *ad-hoc* and WS-BPEL compositions. In this case it is so because we have fixed the number of routes in 10 and varied the number of BGP routers. For 10 routes, the Script MIB has performed worse than *ad-hoc* and WS-BPEL for device composition (look at Figure 10 again). This small difference in the device composition has been propagated, resulting in a more significant difference at the network composition level.

Notice that the performance of WS-BPEL is better than *ad-hoc* in this network composition. Here, this is the result of the native parallel requests issues by the WS-BPEL specification, which is something difficult to be achieved in *ad-hoc* composition because it involves explicit handling of processes and executing threads inside a usually simple code that defines the composition.

These final results make it clear that the final performance of a compositive hierarchy or chains depends not only on the composition technology used, but also on how the elements participating in such a hierarchy are implemented.

VI. Conclusions and Future Work

In this paper we have presented a study on the employment of composition technologies in network management. We have evaluated the use of traditional management technologies such

as the IETF Script MIB, as well as technologies specifically defined to support workflow compositions, like WS-BPEL. Our evaluations have been executed considering a management environment composed of BGP routers that need to be monitored in order to detect anomalies related to the advertisement of BGP routes from remote autonomous systems.

We have considered three main technologies: Script MIB, which is a flexible IETF solution for the deployment of management script on remote managers in a possible hierarchy of managers; *ad-hoc* compositions often implemented on Web systems such as meta-search engines; and WS-BPEL, a recent standard devoted to the specific creation and support of Web services-based compositions.

Previously to our work, it was natural to believe that WS-BPEL – which requires a strong software infrastructure to be deployed – would perform poorer when compared with both Script MIB and *ad-hoc* compositions. However, from our evaluation results it is now evident that the performance issues of WS-BPEL compositions are not as critical as initially supposed. In addition to the performance results associated to it, WS-BPEL also has the advantage of being specifically created for service composition, thus more properly dealing with composition questions, such as native parallel execution support, better design and expressiveness of compositions, and an increasing set of tools available to automate service composition.

This paper concentrated on the performance issues of *ad-hoc* and WS-BPEL service composition for network management contrasting with the Script MIB traditional solutions. Future work of our research will address other aspects of these solutions, such as language expressiveness and scalability, which are as critical as the performance issue.

REFERENCES

[1] Francisco Curbera, Rania Khalaf, Nirmal Mukhi, Stefan Tai, and Sanjiva Weerawarana. The Next Step in Web Services. *Commun. ACM*, 46(10):29–34, October 2003.

[2] Steve Jones. Toward an Acceptable Definition of Service. *IEEE Software*, 22(3):87–93, May/June 2005.

[3] Boualem Benatallah, Quan Z. Sheng, Anne H.H. Ngu, and Marlon Dumas. Declarative composition and peer-to-peer provisioning of dynamic web services. In *18th International Conference on Data Engineering (ICDE'02), 2002.*, pages 297–308, 2002.

[4] Liangzhao Zeng, Boualem Benatallah, Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, May 2004.

[5] Gerardo Canfora, Piero Corte, Antonio De Nigro, Debora Desideri, Massimiliano Di Penta, Raffaele Esposito, Amedeo Falanga, Gloria Renna, Rita Scognamiglio, Francesco Torelli, Maria Luisa Villani, and Paolo Zampognaro. The c-cube framework: developing autonomic applications through web services. In *DEAS '05: Proceedings of the 2005 workshop on Design and evolution of autonomic application software*, pages 1–6, New York, NY, USA, 2005. ACM Press.

[6] Alexandre Alves, Assaf Arkin, Sid Askary, Ben Bloch, Francisco Curbera, Yaron Goland, Neelakantan Kartha, Canyang Kevin Liu, Dieter Koenig, Vikesh Mehta, Satish Thatte, Dannhy van der Rijn, Prasad Yendluri, and Alex Yiu. Web Services Business Process Execution Language Version 2.0, May 2006. http://www.oasis-open.org/committees/download.php/18714/wsbpel-specification-draft-May17.htm.

[7] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. Internet Engineering Task Force (IETF), RFC 3411, STD 62, December 2002.

[8] S. Waldbusser. Remote Network Monitoring Management Information Base. Internet Engineering Task Force (IETF), RFC 2819, STD 59, May 2000.

[9] S. Waldbusser. Remote Network Monitoring Management Information Base Version 2. Internet Engineering Task Force (IETF), RFC 4502, May 2006.

[10] Robbert Van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2):164–206, 2003.

[11] Praveen Yalagandula and Mike Dahlin. A scalable distributed information management system. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 379–390, New York, NY, USA, 2004. ACM Press.

[12] Y. Yemini, G. Goldszmidt, and S. Yemini. Network Management by Delegation. In *International Symposium on Integrated Network Management*, pages 95–107, 1991.

[13] J. Schnwlder, J. Quittek, and C. Kappler. Building Distributed Management Applications with the IETF Script MIB. *IEEE Journal on Selected Areas in Communications*, 18(5):702–714, May 2000.

[14] R. Enns. NETCONF Configuration Protocol. Internet Engineering Task Force (IETF), RFC 4741, December 2006.

[15] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web services Web: an Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing, Vol. 6, Issue 2*, pages 86–93, March/April 2002.

[16] Dimitris Alexopoulos and John Soldatos. Xmlnet: An architecture for cost effective network management based on xml technologies. *J. Netw. Syst. Manage.*, 13(4):451–477, 2005.

[17] Simon St. Laurent, Edd Dumbill, and Joe Johnston. *Programming Web Services with XML-RPC*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[18] A. Pras, T. Drevers, R. v.d. Meent, and D. Quartel. Comparing the Performance of SNMP and Web Services-Based Management. *IEEE eTNSM - eTransactions on Network and Service Management*, 1(2):11, December 2004.

[19] T. Fioreze, L. Z. Granville, M. J. B. Almeida, and L. M. R. Tarouco. Comparing Web Services with SNMP in a Management by Delegation Environment. In *IM 2005 - 9th IFIP/IEEE International Symposium on Integrated Network Management*, Nice, France, May 2005.

[20] ActiveBPEL Engine, 2006. http://www.activebpel.org.

[21] Ravi Musunuri and Jorge A. Cobb. An overview of solutions to avoid persistent bgp divergence. *Network, IEEE*, 19(6):28–34, 2005.

[22] Xenofontas Dimitropoulos and George Riley. Modeling autonomous-system relationships. In *20th Workshop on Principles of Advanced and Distributed Simulation, 2006. PADS 2006.*, pages 143–149, May 2006.

[23] S. Willis, J. Burruss, and J. Chu. Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2. Internet Engineering Task Force (IETF), RFC 1657, 1994.

[24] TU Braunschweig and NEC C&C Europe. Jasmin: A Script-MIB Implementation. http://www.ibr.cs.tu-bs.de/projects/jasmin/, 2003.

[25] R. Neisse, R. L. Vianna, L. Z. Granville, M. J. B. Almeida, and L. M. R. Tarouco. Implementation and Bandwidth Consumption Evaluation of SNMP to Web Sevices Gateways. In *NOMS 2004 - 9th IEEE/IFIP Network Operations and Management Symposium*, pages 715–728, Seoul, South Korea, April 2004.

[26] Ricardo Lemos Vianna, Maria Janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco, and Lisandro Zambenedetti Granville. Investigating web services composition applied to network management. In *Proceeding of the 2006 IEEE International Conference on Web Services (ICWS 2006)*, September 2006. To appear.