Pontifícia Universidade Católica
do Rio de Janeiro

Carolina Valadares

A Multiagent Based Context-Aware and Self-Adaptive Model for Virtual Network Provisioning

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of the Departamento de Informática do Centro Técnico Científico da PUC–Rio, as partial fullfilment of the requirements for the degree of Mestre.

Advisor: Prof. Carlos José Pereira de Lucena

Rio de Janeiro
April 2014

Carolina Valadares

A Multiagent Based Context-Aware and Self-Adaptive Model for Virtual Network Provisioning

Dissertation presented to the Programa de Pós–graduação em Informática of the Departamento de Informática do Centro Técnico Científico da PUC–Rio, as partial fullfilment of the requirements for the degree of Mestre.

Rio de Janeiro, April 09th, 2014

Carolina Valadares

Master's Degree in Computer Science from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) and Bachelor's Degree in Computer Science from the Federal University of Viçosa (UFV).

Bibliographic data

Valadares, Carolina

A Multiagent Based Context-Aware and Self-Adaptive Model for Virtual Network Provisioning / Carolina Valadares; advisor: Carlos José Pereira de Lucena. — 2014.

93 f. : il. (color); 30 cm

1. Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2014.

Inclui bibliografia.

1. Informática – Teses. 2. Sistema Multiagente. 3. Auto-Organização. 4. Conhecimento de Contexto. 5. Gerenciamento de Redes Virtuais. I. Lucena, Carlos José Pereira de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

First, I give thanks to my mother, Maria José Valadares, for being my deepest motivation besides offering the most intimate support. Thank you for the unconditional effort in giving me a promising education. This achievement is ours, mom. To my beloved family, my dear dad, Fernando, and brother, Pedro Paulo, for the concern, support and the unconditional love. I love you all so very much.

To my boyfriend, Hussin, who accompanied me with so much support and understanding. Thank you for the affection and love. To Professor Carlos Lucena, my advisor and biggest reference. Thanks for the support, and for giving great suggestions on how to improve this thesis.

To all my friends and colleagues, specially Nathalia for supporting me. To the LES research group in general for the friendship. To the professors of the committee and staff of the Department of Informatics at PUC-Rio. Finally, to CNPq and PUC-Rio, for funding it.

Abstract

Valadares, Carolina; Lucena, Carlos José Pereira de (Advisor). **A Multiagent Based Context-Aware and Self-Adaptive Model for Virtual Network Provisioning**. Rio de Janeiro, 2014. 93p. MSc. Dissertation — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Recent research in Network Virtualization has focused on the Internet ossification problem (Anderson et al., 2005) whereby multiple independent virtual networks (VN) (Anderson et al., 2005) that exhibit a high degree of autonomy share physical resources and can provide services with varying degrees of quality. Thus, the Network field has taken evolutionary steps on rethinking the design and architectural principles of VN (Blumenthal e Clark, 2001) (Houidi et al., 2008). However, to the best of our knowledge, there has been little investigation into the autonomic behavior of such architectures (Prehofer e Bettstetter, 2005)(Movahedi et al., 2012). This paper describes an attempt to use Multiagent System (MAS) principles to design an autonomic and self-adaptive model for virtual network provisioning (VNP) that fills a gap in the current Internet architecture. In addition, we provide an analysis of the requirements of self-adaptive provisioning for designing a reliable autonomic model that is able to self-organize its own resources, with no external control, in order to cope with environment changes. Such behavior will be required as the next generation Internet evolves. Through our evaluation, we demonstrate that the model achieves its main purpose of efficiently self-organizing the VN, since it is able to anticipate critical scenarios and trigger corresponding adaptive plans.

Keywords

Multiagent System; Self-Organization; Context-Awareness; Virtual Network Provisioning.

Resumo

Valadares, Carolina; Lucena, Carlos José Pereira de. **Um Sistema Multi Agente Auto-Adaptativo baseado em Conhecimento de Contexto para Gerenciamento de Redes Virtuais**. Rio de Janeiro, 2014. 93p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Pesquisas recentes em Virtualização de Redes focaram no problema conhecido como ossificação da Internet (Anderson et al., 2005) , onde múltiplas redes virtuais (Virtual Networks - VN) independentes (Anderson et al., 2005) que exibem um alto grau de autonomia compartilham recursos físicos e podem prover serviços com diferentes graus de qualidade. Nesse sentido, pesquisas na área de Redes de Computadores e Sistemas Distribuídos deram passos evolutivos em repensar o projeto e os princípios arquiteturais de uma VN (Blumenthal e Clark, 2001) (Houidi et al., 2008). Entretanto, até onde sabemos, houve pouca investigação sobre o comportamento autonômico de tais arquiteturas (Prehofer e Bettstetter, 2005)(Movahedi et al., 2012). Sendo assim, esta pesquisa descreve uma tentativa de aplicar princípios de Sistemas Multi-Agentes (Multiagent Systems - MAS) para projetar um modelo autonômico e auto-adaptativo para o gerenciamento de redes virtuais (Virtual Networking Provisioning - VNP). Modelo esse que preenche uma lacuna na atual arquitetura da Internet. Além disso, fornecemos uma análise dos requisitos de um gerenciador auto-adaptativo para projetar um modelo autonômico confiável que é capaz de auto-organizar seus próprios recursos, sem controle externo, para lidar com mudanças no ambiente. Tal comportamento adaptativo será necessário tendo em vista que a próxima geração da Internet está em evolução. Através da nossa avaliação, demonstramos que o modelo atinge seu propósito principal de auto-organizar uma VN eficientemente, dado que ele é capaz de antecipar cenários críticos e executar planos adaptativos correspondentes.

Palavras–chave

Sistema Multiagente; Auto-Organização; Conhecimento de Contexto; Gerenciamento de Redes Virtuais.

Table of Contents

List of figures

List of tables

1 Introduction

The current Internet is a complex, large-scale network which carries a wide range of different services, technologies and applications. Because of its complexity and large scale, trivial management approaches become costly and flawed, as they usually involve human interference when it comes to network update, deployment and simple management tasks (Samaan e Karmouch, 2009). Indeed, there is a consensus that the current Network architecture does not meet current and future requirements. Thus, in order to provide a flexible infrastructure that supports innovations and dynamism in the network, the concept of the Future Internet has been proposed (Fernandes et al., 2011).

Network Virtualization (NV) has recently received substantial attention from the academic community. Because of NV's characteristics, such as the ability to share a single resource among multiple virtual networks (VNs) and the capability of self-management in the face of network degradation, NV's proponents have presented NV as a promising approach to reducing the complexity of managing networks and virtual resources (Blumenthal e Clark, 2001) (Houidi et al., 2008). Beyond the immediate sharing advantage, virtual networks are also very flexible, given that it exhibits and enables different aspects that isolate components through virtualization, which favor, for instance, a self-organization performance. Indeed, different studies in both the Networking and Multiagent areas address the autonomic and cognitive behavior of the Virtual Networks as a promising strategy to reduce the complexity of managing networks and virtual resources.

Thus, it appears that NV represents a key concept in tackling current Internet structural problems, mainly owing to the fact that NV supports the creation of multiple and flexible VNs using a single physical infrastructure. In such a system, there is no interference when running multiple concurrent VNs and each VN is capable of managing itself and its own resources in the face of interference from the surrounding environment. In other words, VN is a new architectural concept in which each VN runs its own protocols, services and technologies (Marquezan et al., 2010), and responds to users' requests just as any traditional network architecture. In addition, VN also supports autonomic self-management through adaptation plans which allows the efficient distribution of physical resources among VN's virtual devices.

Many studies in the Network field (Marquezan et al., 2010) have applied Multiagent (MAS) and Self-* approaches to support solutions to Network Virtualization management. This research, however, discusses management issues that arise in a VN infrastructure from a MAS perspective. In this sense, the virtual network provisioning or adaptive maintenance system aims at dealing with dynamic changes caused by variations in the physical and virtual networks with a view to producing efficient use of physical resources and a high level of service. These changes are related to failures, mobility, migration and maintenance needs.

We intend to deal with the additional complexity of this new VN concept by enabling autonomic and self-management behavior through Multiagent and Self-* paradigms. Thus, the main goal of this research is to fill in the gap that virtualization studies exhibit by covering design principles, from the autonomic computing perspective (Tesauro et al., 2004), in terms of autonomic distribution, means of communication, degree of intelligence and autonomy. We aim to explore self-organization with VN by combining it with Network research to enable proper management of such systems beyond offering a higher degree of autonomy and intelligence.

The focus of this Master's Thesis is strictly managerial in terms of resource usage and network stability. In order to achieve a high degree of independence and exhibition of local property, in which local solutions lead to local consequences, our proposed management model comprises of several agents spread all over te networks. Thus, the virtual and physical nodes embed autonomous and intelligent agents, which will exchange messages and cooperate with each other to carry out distributed VN management. Depending on the Network condition (link health/resource availability), the proposed model anticipates a future critical scenario and triggers specific adaptive plans to keep the VN running based on predefined requirements. Critical scenarios refer to link/router degradation and the adaptive plans refer to management solutions for routers and links.

Accordingly, our main motivation is to increase the degree of autonomy, decrease the impact of self-adaptation, while reducing the need of message exchanges to resolve any critical event. Moreover, we rely on the notions of agents, organizations, cooperation and their decentralized and pro-active nature to ensure distributed negotiation and synchronization between the substrate nodes and virtual resources. This, in turn, enables a large-scale autonomic self-* environment. We observe, however, that one of the major challenges in this domain is the union of the variety aspects of the same research domain and the lack of studies in the self-* features and autonomic management for a given Virtual Environment.

As our proof-of-concept, we implemented and validated, through testbed experiments, a Virtual Network Provisioning System (VNPS) wherein the VNPS acts upon a critical scenario by re-organizing itself. First, analyze and clarify the autonomic network management concepts that lead to self-organization as well as how the choice of the right communication approach in a fully decentralized environment better supports the efficient emergence of self-adaptive events. Then, we showed that, in the case of environment degradation, our model is able to anticipate a possible future failure. Such a failure because of the lack of physical resources, triggers an adaptive event and executes the adaptive plan in a matter of seconds. We also show that for larger networks with increased complexity, our model is still efficient, executing adaptive plans by exchanging a small number of messages, which increases in a linear fashion. Finally, we show, through the VNPS validation, a scalable and robust way to evaluate the effectiveness of the self-organizing system, as well as to self-configure its virtual resources in critical scenarios.

Note that, we believe that the union of this proposed model with recent advances and trends in the Network field is a great step to enable autonomic Virtual Networks, as the Future Internet requires (Egi et al., 2007).

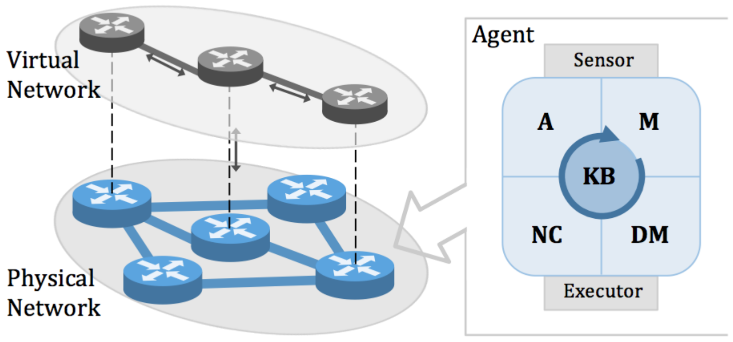


Figure 1.1: Virtual Network Architecture and the agent autonomic control loop overview

1.1 Problem Statement

A strong candidate for the Future Internet architecture, design and principles is the Networking Virtualization. In order to fill the gap the current Internet architecture exhibits, Virtual Networks (VN) have been the main key to providing a more autonomic and dynamic Internet architecture which exhibits self-properties. A VN represents the virtualization of an entire network architecture and its associated applications and services on the top of a single physical infrastructure. Moreover, VN also enables the deployment of multiple independent virtual networks that run simultaneously and completely isolated from each other. Furthermore, an administrator can ensure complete utilization of one available virtual network to run its own services, applications and management solutions.

Existing work on VN focus on different branches of the same domain. They are as follows:

- Virtualization technologies, techniques and mechanisms of virtualization (Fernandes et al., 2011) (Pisa et al., 2010); in which they show advantages and limitations of the virtualization technologies and tools for creating multiples virtual environments.
- Virtual Network Mapping (Cheng et al., 2011) (Houidi et al., 2008); which is responsible for mapping virtual routers and links into the physical resources.
- Live virtual migration of virtual machines (Houidi et al., 2008); which optimizes the live migration of virtual machines across the physical infrastructure;
- Autonomic resource management of the physical and virtual networks and their components (Ruth et al., 2006) (Marquezan et al., 2010) ; which is responsible for managing the efficient use of physical resource on network virtualization.

Assuming the existence of as established virtual network and disregarding the impact of virtualization tools and techniques, we are specifically interested in the autonomic treatment of physical and virtual resources while managing the efficient use and distribution of physical resources among virtual devices. As a result, the ideal virtual network provisioning model has to face environment changes such as high network flow variations and unbalanced usage of physical resources, in order to maintain a stable network load and a high degree of quality of services while self-managing resource assignments and reassignments. Hence, enabling self-organizing mechanisms is the biggest concern of this master's thesis while dealing with the management of such an architecture.

1.2 Objectives

Our objective in this Master's thesis is to propose a feasible solution for improving self-organizing behavior of the virtual network management model. We also aim to distribute the management responsibilities among autonomic agents spread through the physical and virtual networks. Our solution aims at improving the performance of adaptive behaviors while at the same time keeping the incurred number of messages exchange, and, therefore, its overhead, to minimum. To this end, our approach consists in:

- Differentiating the critical scenarios so the degree and type of criticalness of resource availability can be mapped into adaptive plans;
- Providing mechanisms to autonomically reassign physical resources among virtual devices to face environment changes.

It is also our objective to take advantage of previous VN management proposals and autonomic network progress regarding self-organization, self-awareness and normative rules.

1.3 Contribution

Our main contributions are as follows:

1. We analyze and present intrinsic properties of the VN domain that favour self-organizing behavior.
2. We propose adaptive plans to be automatically triggered in the case of three different critical scenarios: (i) Virtual Router overload, (ii) Virtual Link overload and (iii) Physical Router overload.
3. Our approach improves the rate of programmability of the autonomic control loop through enabling the self-tuning feature. Such a feature improves the degree of autonomy while decreasing the need for embedded knowledge and setting parameters in each autonomic agent.
4. Our mechanisms can be incorporated in existing self-organizing systems that use the concept of autonomic control loop as the main tool to enable self-adaptive behavior.
5. We evaluate our approach through real test-bed experiments. We also evaluate it from a simulation based on real behaviors to show the effectiveness of our techniques in terms of (a) efficiency in triggering

different adaptive plans depending on different environment changes and (b) performance gain in terms of number of message exchanges, and therefore overhead.

1.4 Master's Thesis Organization

The rest of this thesis is organized as follows:

Chapter 2 describes the state of the art on the following topics that are addressed on this master's thesis: network virtualization, autonomic network management, autonomic computing, self-* properties and norm-based models. In addition, the relationship among these topics is also analyzed regarding the domain of virtual network management that comprises self-* properties. It also gives a detailed description of XEN, the virtualization tool used for the development and validation of the proposed model.

Chapter 3 provides a literature survey as related work. Three topics are addressed in this chapter. First, the conditions made in Network virtualization are presented. Second, it is presented a description of the virtual network management solutions. Finally, terms and concepts used in context-awareness, multiagent systems, self-organization and norms are defined.

Our proposal from a self-organizing perspective is discussed in Chapter 4, in which we describe the main self-organizing properties present on the VN domain. Chapter 5 gives a detailed discussion on how self-awareness is coupled to this proposal, while, in Chapter 6, we present the autonomic control loop extension by the use of the normative concept. Both self-awareness and normative autonomic control loop concepts enrich the proposed model by enabling a more flexible, programmable and autonomic system.

Chapter 7, in turn, is dedicated to the experiment setup and results. In this section, we summarize the main differences regarding the efficiency between our solution and centralized solution. Finally, we conclude the thesis and present some limitations of our work as well as possible future work in Chapter 8.

2
Background

Recent research studies have pointed out that providing a distributed self-organizing approach for the management of virtual networks is a viable solution to deal with the increase of complexity that network virtualization has brought. We strongly believe that this complexity could be handled by autonomic computing together with the concept of Multiagent System (MAS), Norms, and Self-* properties (which includes self-adaptation and self-awareness).

In this chapter, we first provide an introduction of Network Virtualization, which is the domain of this research project. Next, we define some concepts involved in our proposed solution. In particular, we briefly describe some definitions and advantages of (i) Multiagent System (MAS), (ii) Self-organization, (iii) Self-Awareness for MAS, and (iv) Normative models and Reputation. In the end, we provide some preliminaries from mechanisms used by virtualization tools, specially XEN, which is the virtualizing tool used for the design, development and experiments setup of this project.

2.1 Network Virtualization

Network virtualization has been taken as a promising technique to deploy future networks that meet current and future users requirements (Blumenthal e Clark, 2001) (Clark et al., 2004) (Fernandes et al., 2011). The main idea behind network virtualization is of slicing (sharing) physical resources to create multiple virtual networks capable of running its own protocols, services and management solutions. Hence, its main concept relies on the fact that it adjusts the network flow and routes, in an autonomic way, dismissing any kind of external control. It aims to maintain the quality of service (QoS) defined in the SLA by controlling the agent's behaviors and adapt itself in order face environmental changes.

The main components of the architecture for virtualization are (i) virtual routers (virtual nodes), and (ii) physical routers (physical/substrate nodes). In this sense, a virtual network represents a collection of virtual routers connected together by a set of virtual links to form a virtual topology. Thus, the virtual network is essentially a subset of virtual nodes mapped on the top of the underlying physical network. The virtual node is hosted in a particular substrate node, which means it is a virtual slice of the physical host, comprising CPU, memory RAM, storage capacity, operating system, and so on. On the other hand, the substrate node is composed of real physical resources (CPU, memory RAM, storage, etc.) and holds virtual nodes and virtual links, as depicted in Figure 2.1. The physical network consists, for instance, of devices such as routers, access points or physical links, and is able to embed many virtual devices. Further details regarding virtual network architecture can be found in projects like 4WARD (Architecture and Design for the Future Internet, 4WARD FP7 project).

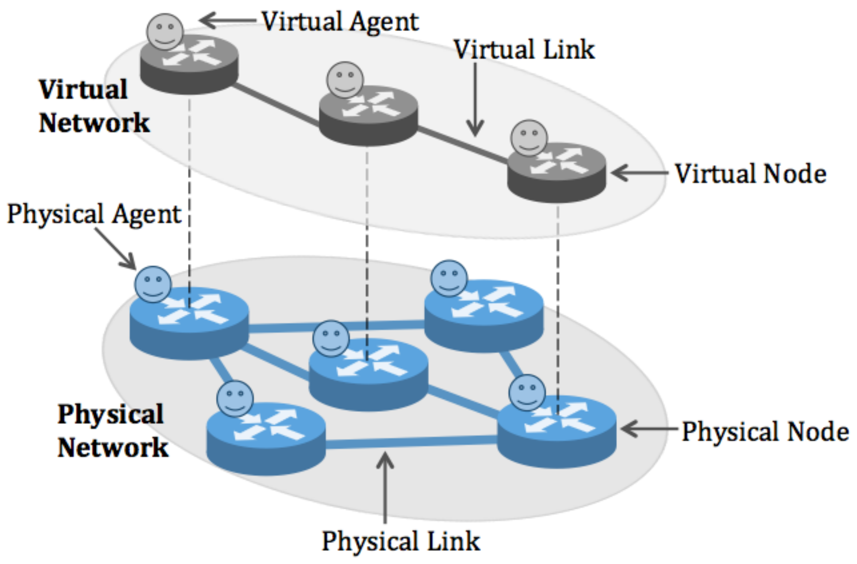


Figure 2.1: Virtual Network Architecture

In the case of an autonomic management solution fo virtual networks, each virtual and physical router might hold a resource manager, the physical and virtual network management component.

2.1.1 Virtual Network Model

Physical Network

To formalize, a physical network might be denoted by $PN = (PR , PL)$, where PR is the set of real routers and associated with each node, PR , in this set is some routers attributes, such as CPU capacity and Memory RAM, among others. The set PL, in turn, is a set of real links and each link, PL, has an associated bandwidth capacity, the delay of that link and other link attributes.

Virtual Network

A virtual network consists of a desired topology in which a set of virtual routers is mapped into a set of physical routers. In this master's thesis, we denote the virtual network as the graph $VN = (VR\ ,\ VL\ )$, where VR is a set of virtual routers and associated with each virtual node is a set of parameters that the embedding manager should satisfy. These parameters are related to the required CPU capacity, Memory RAM, bandwidth capacity associated to each network device, etc. Similarly, VL is a set of virtual links and associated with each virtual link is a set of parameters related to the link setup. The embedded manager should satisfy these parameters in which, for instance, can be required bandwidth capacity, maximum delay, maximum loss rate etc.

Router & Link Stress

Router and link stress are important factors in quantifying the resource utilization. The Network research community has used the concept of resource stress in previous literature (Zhu e Ammar, 2006) (Yu et al., 2008). Router stress is defined as the fraction of all the CPU capacity of the substrate node that is mapped to some virtual node. Similarly, the link stress is defined as the fraction of all the bandwidth capacity of a substrate link that is mapped to some virtual link.

We define the capacity of a link or a router in terms of its stress. For a router, for instance, the stress is defined as the total CPU capacity minus its router stress while for a link its the total bandwidth capacity minus its link stress.

2.1.2 Autonomic (Virtual) Network Management

Many current Networking system are still managed by human operators. Most Network architectures show as a major limitation the high dependence on external control. For instance, simple operations, such as the introduction of a new network device, require the overhead of taking the system down and manually introducing changes such as reconfiguring its relation to other variables and monitoring programs. Consequently, the performance of these managed networks is strongly constrained by the expertise of human operators.

Because of the increase of complexity and scale of these architectures, the design of an efficient and scalable mechanism to function management taks is needed in order to alleviate the responsibilities of such human operators. To this end, technologies such as mobile agents (Magedanz e Karmouch, 2000), active networks (Boutaba e Polyrakis, 2002) and policy languages (Calo e Sloman, 2003) were proposed to automate the deployment of different management strategies for performing the necessary management operations.

However, the interference of human operators to perform management functions is still highlty demanded. Even though the Network may exhibit a degree of autonomy, the operators still have to precisely describe and program their desired behavior, enable necessary policies and, to continuously modify this behavior in response to changes in the environment. Another main limitation of these automated-management approaches was the inability to evolve and adapt with changes in either supplied business objectives or users' requirements.

Thus, to overcome this current Network impasse, also known as the Internet *ossification* problem, and proceed with an autonomic network management diversification proposal, the Network Virtualization concept has gained the confidence of a wider research community. In this sense, Autonomic Network virtualization management involves autonomic operations such as instantiating, deleting, monitoring, migrating virtual network elements and setting its resource allocation parameters, all in a completely independent manner. Such functionalities are what make the management system a suitable model for creating and managing multiple virtual networks and, as a consequence, supporting the pluralist approach for the Future Internet. This is evidenced by its ability to create multiple customized virtual networks at the same time it exhibits a flexible management and a real-time control (Fernandes et al., 2011). An important challenge on network virtualization is the efficient allocation of the physical resources at virtual network mapping and adaptive provisioning stage. To accomplish such efficient use, the management of the physical resources should be frequently executed at runtime in order to deal with the variation on the load requests of different users.

2.2 Multiagent System

An agent is a virtual entity situated in an environment that changes over time. Through its sensors, the agent is capable of perceiving its environment, and through its effectors, it is capable of performing actions that affect the environment. Agents group together and form multiagent systems, also known as societies of agents, where they work together to solve problems that are beyond their individual capabilities (O'Hare e Jennings, 1996). On the basis of its own knowledge, an agent may decide to regularly act upon an environment, without the user having explicitly controlled that (Serugendo et al., 2004). In order to build a partial representation of its environment, which constitutes its knowledge, agents interact with each other, and with their environment. (Serugendo et al., 2004) also state that the interaction dynamics between an agent and its environment lead to emergent functionality, even though no component is responsible for producing a global goal.

Thus, multiagent system is characterized as a virtual system composed of multiple interacting autonomic entities within an environment. The agents may carry a degree of intelligence or may be completely reactive. Multiagent systems are often used to solve problems that require distributed autonomous actions among cooperating entities, features that are usually difficult to enable in an individual agent or monolithic system. The MAS field contains many branches of research studies. In this research, we are especially interested in the concept of self-organizing systems, where each agent is governed by a set of constraints or norms, so-called self-properties.

We propose a virtual network architecture applying the MAS paradigm as a modeling foundation. We have chosen such a paradigm mostly because it seems to be particularly suitable to build autonomic systems, due to properties of agents, such as autonomy, proactivity, adaptability, cooperating, and mobility. Moreover, the notions of agents and organizations and their decentralized and pro-active nature match well the requirements of large-scale autonomic computing environments.

Accordingly, this project provides the design and evaluation of a distributed system based on MAS to take advantage on the distributed negotiation and synchronization between the substrate nodes and virtual resources through their agents. This enables the virtual and physical nodes to be able to handle autonomous and intelligent agents, which exchange messages and cooperate to each other to carry out the distributed virtual network management. Hence, we apply such concepts to enable communication between the substrate and virtual agents to improve the performance and scalability of the distributed and autonomic virtual network manager, in order to tackle the virtual network adaptive provisioning challenge.

2.3
Self-Organization, Self-* properties & Self-Awareness

(Serugendo et al., 2004) states that self-organization essentially refers to a spontaneous, dynamically produced (re-)organization. A self-organizing system functions without any central control, and through contextual local interactions, the autonomous agents perceive its local environment to draw conclusions about it to act accordingly. Individual components achieve a simple task individually, but a complex collective behavior emerges from their mutual interactions. Thus, self-organizing systems are able to modify their structure and functionality to adapt to changes to the requirements and to the environment based on previous experience.

The paradigm of self-organizing systems in the field of multiagent systems is an organization where the agents operate with a high degree of autonomy without central control, and function based on local interactions and views (Serugendo et al., 2006). Self-organization takes advantage of the intrinsic characteristics that multiagents offer, such as a high degree of distribution, unsynchronized interactions and cooperative behavior among agents. These properties enable spontaneous re-organization of agents and resources in case of a change in the environment. The re-organization is an intrinsic feature of a self-organizing system and results from internal constraints or mechanisms, triggered by local interactions between its components (Camazine et al., 2001).

In summary, self-organization can be defined as the emergence of system-wide adaptive structure and functionality from simple local interactions between individual entities. And, regarding the proposed model, it brings the ability to self-manage its own resources to cope environment changes in order to meet polices and user's requirements.

Self-awareness

Self-* properties and autonomic behavior arise from the capability of the system to internally and autonomically identify any environment changes without human administrator interference. Self-awareness, in turn, requires *sensing* capabilities and triggers *reasoning* and *acting* behaviors. Self-aware systems are currently modeled with autonomic monitoring, planning and plan execution capabilities at the level of the autonomic managers. Awareness is enabled in self-organizing systems through the ability of sensing their environment in different ways, and take decisions accordingly. The capability of sensing the environment is given through the process of knowledge acquiring and knowledge sharing.

2.3.1 An Abstract Self-Organizing Model

When it comes to Self-organizing systems, to the best of our knowledge, the most common and most widely cited self-organizing reference model is the autonomic control loop (ACL) proposed by IBM and called MAPE-K model (Horn, 2001) (IBM, 2005), as shown in Figure 2.2. Based on notions of (M)onitoring, (A)nalyzing, (P)lanning (also Decision Making), (E)xecuting decision and a (K)nowledge base, MAPE-K has become a base model from which it is possible to extend more advanced and sophisticated self-organizing systems. From these components, we can conclude that the autonomic manager achieves a higher degree of autonomy and self-organization through the autonomic monitoring and decision-making process. The management tasks involved in the MAPE-K model are related to: (i) monitoring of the managed components by capturing necessary unstructured measurement of the environment which are of significance to the self-properties of the environment; (ii) analyzing the collected data, translating and aggregating the unstructured and brute data into local knowledge, which is then stored in a knowledge base; (iii) planning adaptation actions depending on the environment variation; and (iv) executing the decided plans upon the environment. We highlight that the adaptive plans executed by the model may or not depend on collaboration with other agents, even though it is completely autonomic and does not require any external administrator intervention.

An adaptation is normally triggered in response to changes or high variations of the environment. However, it may also be triggered periodically or in response to external events. Hence, the presented self-organizing model is flexible enough and, depending on the system' design, it exhibits a specific adaptation behavior. In fact, some factors impact the adaptation rate, sensibility and autonomy of the model. Thus, the more dynamic and flexible the environment is, more stable and less reactive the adaptive model should be, and yet, the more autonomic less dependent on external and internal support and control.

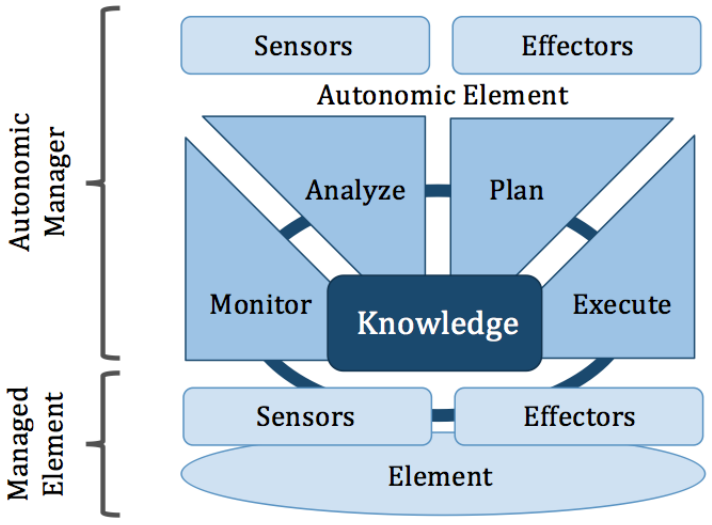


Figure 2.2: Autonomic Control Loop (ACL)

2.4 Norms & Reputation

Norms constitute a coordination mechanism among heterogeneous agents. One of the challenges regarding norms in multiagent systems is the design and implementation of distributed environment in which coordination must be achieved among self-interested agents. Norms can be used for this purpose as a means to regulate the observable behavior of agents as they interact in pursuit of their goals (Axelrod, 1997) (Dignum, 1999).

In this sense, recent research efforts have used the concept of social norms as a mechanism to regulate and control virtual societies, especially those composed of heterogeneous environments using different types of agents. In terms of multiagent systems, the notion of a normative system with each agent being regulated by its own behavior or norm supports distributed virtual environments especially in situations where having a central authority is not feasible. Having a central entity responsible for preventing a given undesirable behavior by constraining the architecture does not scale well. Thus, in highly distributed environments, the responsibility of enforcing behavior through societal norms is distributed over the members of the society of agents.

A normative multiagent system is the union of the concepts of multiagent and norms. We are dealing with a distributed community of heterogeneous agents together with a normative system to control the autonomic behaviors exhibited by the model while preventing the system from malfunctioning. Each agent can decide to follow a set of related norms or face the consequences of declining them. Most research work in norm-based systems for a heterogeneous agent environment (Boella e Torre, 2004) (Boella e Torre, 2005) identify norms as having obligations, prohibitions and permissions. These three properties determine how an agent makes decisions when they are subject to norms. Such normalization has been widely used (López et al., 2002) (Lopez et al., 2004) to deal with coordination and security issues arising from multiagent systems.

In this thesis, we join the concepts of multiagents and norms to enrich the self-organizing model. We apply the concepts of norms to self-tune the model's constraints and thresholds in order to control better the adaptive behaviors of the self-organizing model based on the local environment. The main idea is to enable different sets of norms depending on the context and variations of the surrounding environment. Through norms, we are able to use a self-organizing core capable of self-configuring and self-adapting its own parameters and thresholds based on current needs.

Note that, in what follows, we could use the Z language (Spivey, 1989) throughout the thesis to specify a formal model of the normative autonomic control loop, the main concept behind norms to support a more autonomic self-adaptive behavior. Z is based on set-theory and first order logic, with details available in (Spivey, 1989). However, we have chosen to show few examples of the predetermined attributes in Z with the remainder of the norms descriptions in natural language so as to make them easier to read.

2.4.1 An Abstract Normative Model

We deal with the normative concept to enable self-tuning in self-organizing models. In a dynamic and highly distributed environment, agents are normally required to collaborate in order to achieve the desired common goal. Because of the heterogeneity of the surrounding environment, some interests of different agents may exhibit conflicts (Zambonelli et al., 2011). Thus, in order to avoid conflicts that may appear among these autonomous agents we enable the concept of norms.

The concept of norms also presents a particular relevance when it comes to controlling and ensuring the proper functioning and adaptation of a complex environment composed of autonomous components. To this end, norms are applied to collectively regulate all activities and behaviors of the managed components (Tinnemeier et al., 2009) (García-Camino et al., 2009), and, depending on the system' design, specific components are responsible for enacting and controlling the prescribed norms. Moreover, the definition of proper internal or external components and mechanisms to control and limit the system behavior and prevent malfunctions is flexible and depend mostly of the system design.

There are some questions raised on research studies related to norms and its incorporation in multiagent systems. Thus, since norms are explicitly represented and are interpreted as a special set of constraints of the domain, where all agents of the same environment may tie to, norms can be designed, represented and controlled depending on the domain in which they occur. Therefore, how norms are represented, how the modeled system monitors the modeled agents behavior, enforcing sanctions and rewards, in case of violations and good behavior respectively, and how norms evolve are open questions that tend to be answered depending on the domain study.

As for the case of the self-organizing model studied in this research, we tend to express the normative rules to drive the behavior of the managed elements and their adaptation constraints depending on the environment variation. Indeed, these normative rules enable the building of a set of self-adaptive patterns that are well supported in various scenarios and environment condition.

2.5 XEN: A Virtualization Tool

To validate the virtualized architecture we have to choose an operating system virtualization to perform the roles of multiple independent virtual nodes on a shared substrate. Research studies on the network virtualization field (Egi et al., 2007) (Pisa et al., 2010) have pointed out XEN networking tool as an enabling technology that has reached a certain maturity over its competitors. It is a Virtual Machine Monitor that makes the simultaneous running of multiple operating systems on a single system in a complete and decentralized way. This represents a considerable advantage for the context of our self-organizing model. XEN may be considered complete as it supports full virtualization by allowing unmodified operating systems being hosted under the physical machine, and it may be considered decentralized as there is no central entity responsible for controlling and managing the virtual machines and its services. In other words, the virtual machines hosted by XEN are independent and do not influence each other.

The XEN Platform consists of:

1. *XEN Hypervisor*
   It is a virtual machine monitor, which is responsible for providing services to allow multiple computers operating system to run on the same physical machine concurrently. It is located above the physical hardware and coordinates the low-level interaction between virtual machines and physical hardware.

2. *Domain0*
   It is the virtual machine host environment, also referred to as controlling domain, which is a special domain that provides the management environment. Thus, it manages the virtual machine host components and its virtual machines.

3. *Virtual Machines*
   Also known as Guest Domains, User Domain or simple Dom U. It holds a virtual disk that contains a bootable operating system, a Virtual machine configuration information, which can be modified through Virtual Machine Manager and a number of network devices, connected to the virtual network provided by the controlling domain.

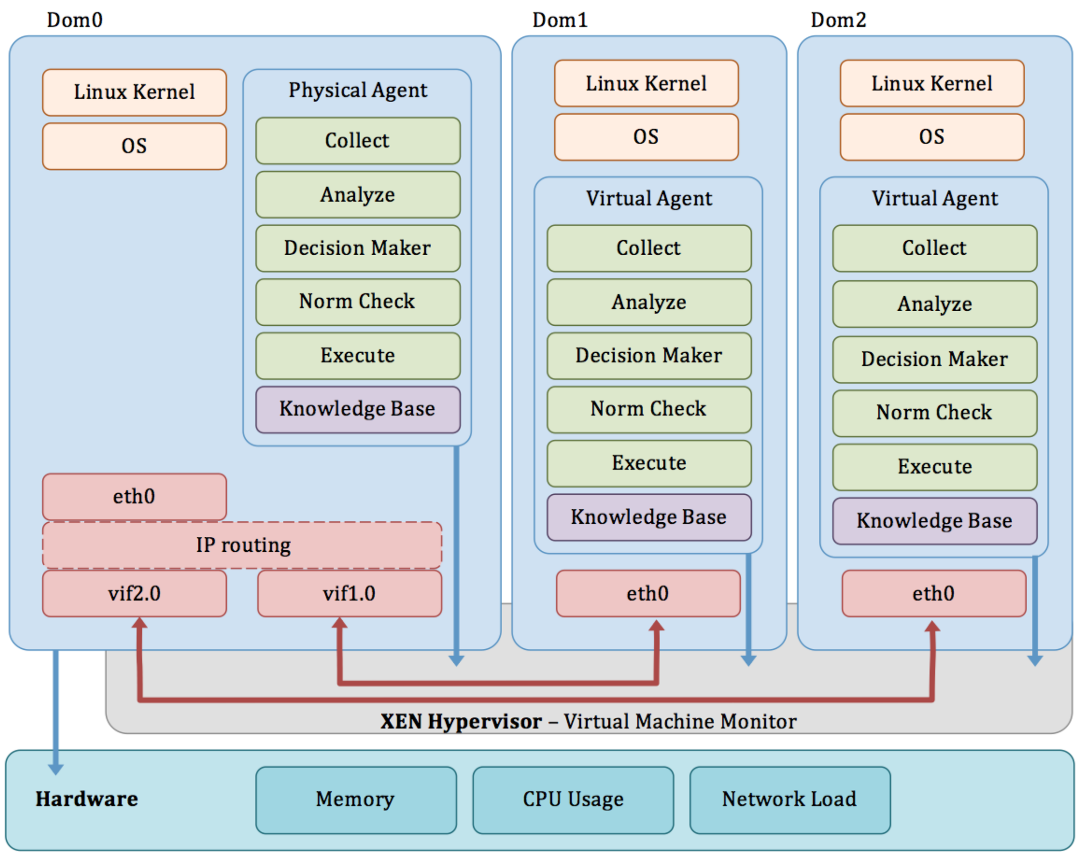


Figure 2.3: Physical machine (*Dom0*), Virtual machines (*Dom1 and Dom2*) and XEN Networking routed.

As shown in Figure 2.3, the XEN hypervisor represents an intermediate layer between the physical machine's hardware and the guest domains. It intermediates the guest domain access to the physical machine's hardware through the Domain 0, a driver domain responsible for the communication itself. Domain 0 differs from the other guest domains in that it has special privileges and total access to the hardware of the physical machine, providing, this way, a reliable and efficient hardware support (Egi et al., 2007). The guest domain, as a consequence, is capable of running privileged instructions as it were real machine through the Domain 0.

Virtual machines (DomUs) are separated and isolated from each other, so that the execution of one does not affect the performance of the others. There is no way but through a private network to establish communication between them, even among those who share the same physical substrate.

2.5.1 XEN Networking

XEN provides mainly two well-known packet forwarding mechanisms: Bridging and Routing. Both refer to the technique to move packets between the real interface and the guest domains. In (Egi et al., 2007), authors have evaluated and compared the performance of these two routes forwarding plans inside the XEN virtual machine monitor environment, Bridging and Routing, aiming to identify design issues in Virtual Routers. Moreover, the authors have shown that the bridged scheme used in XEN for interdomain packet transfer has way more significant costs for Dom0's forwarding performance.

For instance, (Egi et al., 2007) outlines that with Bridging mechanism all the Guest Domains are able to share the same network, which may offer an increased complexity of the driver domain. Because it uses software bridges within the driver domain, dom0, to transfer the data packets between the real interfaces, residing in Dom0, and the virtual interfaces associated to the DomUs. While in Routing mechanism, data packets forwarded from one to another physical interface in Dom0 will transverse the same path as it would in native Linux. In other words, it eliminates, from dom0 forwarding path, the channels through the hypervisor as well as the bridges attached to each physical interface. Figures 2.5 and 2.4 depict this difference.

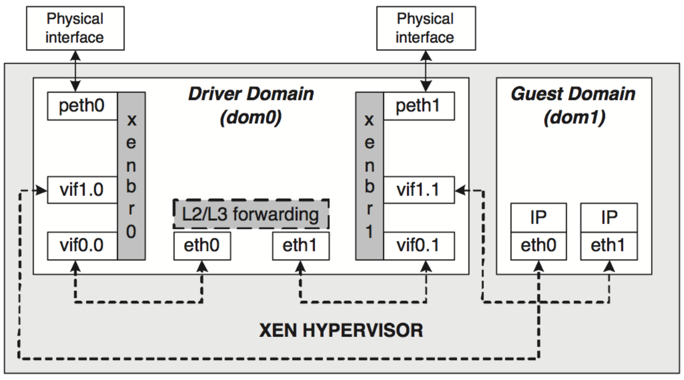


Figure 2.4: Xen's bridged network internals (Egi et al., 2007)

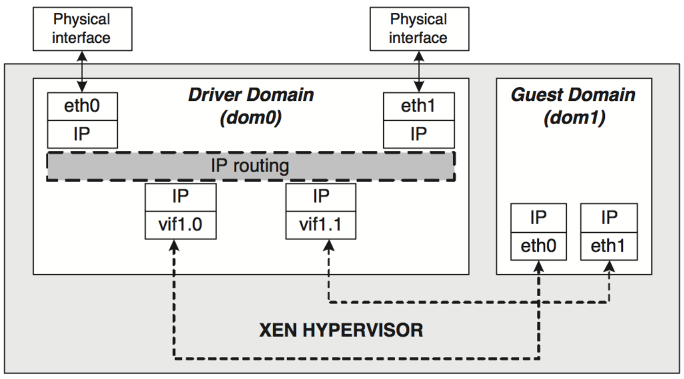


Figure 2.5: Xen's routed network internals (Egi et al., 2007)

Being aware of the test scenarios evaluated in (Egi et al., 2007), we have implemented our autonomic architecture using XEN Routed Networking, as shown in Figure 2.3. We created a virtual network consisting of two main private networks: External and Internal, in which the guest domains (Dom1 and Dom2) are hosted in a different internal private LAN, and their traffic data are routed to the external private network, outside of Dom0, the driver domain.

For further details regarding Xen creation and live migration of virtual machines, refer to Appendix A.

3
Related Work

In this chapter, we present an overview of the existing research studies addressing the VN provisioning problem and that are related to this domain. We first categorize previous proposals of relevant projects on the domain of network virtualization that have an impact on and are of interest of this Master's project. Next, we describe some advances in autonomic computing, where we explain the assumptions made by those studies regarding self-* properties for VN domain and their respective impact. Finally, from a MAS and Self-* perspective, we explore some state-or-art research on paradigms and applications for self-organizing and norm-based systems.

3.1
Virtual Network

The problems involving the concept of Virtual Network are widely addressed in various areas of research. Academic and industry studies aim to resolve different aspects of Virtual Network Provisioning and Management problem from several perspectives from either the Network or the Multiagent fields.

There are studies related to VN deployment, concerning the resolution to virtualize a Network and its components, by comparing several virtualization tools, and virtualization approaches. The authors of (Fernandes et al., 2011), for instance, define functionalities needed to build a virtual network architecture and identify the overhead incurred by some virtualization tools by comparing them with native Linux. Similarly, (Egi et al., 2007) evaluate the performance of the router forwarding plane inside the XEN environment with a view to identifying design issues in Virtual Routers while (Pisa et al., 2010) compare virtualizing tools such as XEN and OpenFlow in terms of migration models. In particular, in terms of architecture and structure, which covers, for instance, the capability of a single infrastructure in deploying Virtual networks, some research studies attempt to answer such problems regarding physical and virtual networks, mainly involving concepts such as tolerance and limits. The concept of situation/context awareness is also presented in projects such as 4WARD (Architecture and Design for the Future Internet, 4WARD FP7 project), in which the contextual view provides the basis to make a proper decision, based on the local and global state of the network. Further details regarding virtual network architecture, self-organizing and self-awareness, from a mixed perspective of Network and Multiagent areas can be found on the 4WARD project (Architecture and Design for the Future Internet, 4WARD FP7 project).

We also find research on the VN domain in a higher level. For instance, some are related to the provisioning of the virtual network itself. In this sense, an important branch of this problem is related to the mapping of virtual nodes into physical substrate at the moment a virtual network is requested. Upon a request, the mapping problem attempts to best fit a virtual topology over the physical infrastructure (an attempt to approach an optimal solution), being then responsible for providing a Virtual Network upon a VN request. This problem presents a high complexity in finding an optimal mapping, as it is known as an NP-problem. Therefore, this area has been widely addressed in a varied range of studies, such as (Cheng et al., 2011) (Houidi et al., 2008).

In both research, the authors tackle the problem of mapping virtual resources in the physical infrastructure, concerned about the efficient resource mapping while dealing with the simultaneous optimization of the placement of virtual nodes and links on a substrate network. The difference, however, relies on the fact that (Cheng et al., 2011) proposes a central approach to solving the mapping problem, though network topology awareness, and (Houidi et al., 2008) comes up with a self-organizing solution, in which it embeds agents inside every physical node in order to distribute the responsibility to solve the mapping problem. The authors of (Houidi et al., 2008) states that a distributed solution is necessary since a centralized solution suffers from scalability.

3.2 Dynamic VN Management

Assuming that the virtual network has been provided, the adaptive maintenance itself comes into play in order to deal with dynamic changes from the variations in the substrates and virtual networks. Such changes are related to failures, mobility, migration and maintenance needs. The idea behind the adaptive provisioning is to maintain the original topology and service levels agreements during the virtual network lifetime. The virtual network provisioning involves virtual routers and links management, such as live migration of routers, and virtual router reallocation.

Although there are in the literature substantial amount of work dealing with Virtual Network Mapping, from the Network perspective, to the best of our knowledge, there are a few studies on adaptive provisioning of instantiated virtual networks to cope with dynamic changes in service demands and resource availability, mainly from the MAS perspective. In order to solve the virtual network provisioning problem, few approaches have been suggested, dealing mostly with (i) virtual node live migration to a distinct host and (ii) virtual link reassignment and setup to preserve the virtual network topology.

Thus, there are research studies addressing the feasibility and efficiency of enabling online updates on the Virtual Network environment. From a management perspective, research such as (Kamamura et al., 2011) (Miyamura et al., 2011) propose models to deal with the autonomic and self-organizing behavior of the environment provided by virtual networks. We highlight that some are concerned in finding the right balance between autonomy and Self-adaptation, while others are concerned in exploiting some of the varied range of adaptation behaviors. In addition, (Houidi et al., 2011) addresses the provisioning of virtual resources in future networks relying on the Infrastructure as a service principle.

Moreover, the authors of (Ruth et al., 2006) proposed an autonomic system called Violin, which manages a virtual environment, composed of virtual nodes capable of live migration across a multi-domain physical infrastructure. It addresses the live migration problem considering different topologies and even distinct physical networks. Similarly, (Houidi et al., 2008) also deal with the distributed management issues involved with live migration of virtual routers. This management concept involves, for instance, applying some multiagent techniques in which distributed agents take over the management activity itself. Furthermore, (Marquezan et al., 2010) proposes a distributed self-organizing model to manage the substrate network resources. The authors of (Senna Daniel M. Batista e Madeira, 2011), on the other hand, deal with virtual link reassignment in the case of unbalanced use of links in a virtual network, where it changes the mapping of virtual links if the load of specific physical links increases more than a certain threshold.

Furthermore, the authors of (Houidi et al., 2010) propose a distributed self-organizing model based on MAS properties to manage the substrate network resources in the case of failures and link degradation. In the proposed solution, they apply the concept of agents spread all over the physical infrastructure, in which each agent is in charge of controlling the physical router and also all virtual routers that this physical entity may embed. This paper highlights Self-* properties and briefly describes how the autonomic entities communicate and cooperate to maintain the virtual network stability and high levels of functionality.

Although all these proposals provide specific solutions for the virtual network provisioning and control problems, most of them do not take into consideration the management activity from a fully distributed perspective. Therefore, we note that these approaches have treated the virtual network management from a semi-decentralized way, in which the autonomic entities are spread only over substrate nodes. Differently from those highlighted research, the self-organizing model proposed in this project addresses the management of substrate and virtual resources by taking advantage of the total distribution of the autonomic entities spread all over the network, including virtual networks rather than only substrate nodes.

3.3 Autonomic Computing & Networking

In the Autonomic Networking field, we came across research that deal with Self-* paradigms applied on autonomic networks, such as the works of (Prehofer e Bettstetter, 2005) and (Samaan e Karmouch, 2009). The authors of (Prehofer e Bettstetter, 2005) propose four self-organizing paradigms to be applied in autonomic networks, while the authors of (Samaan e Karmouch, 2009) describe a survey covering all possible self-organizing and self-aware properties that a VN may exhibit; they also cover an overview of knowledge retrieving and sharing.

The authors of (Prehofer e Bettstetter, 2005) propose design paradigms for developing a self-organized network function. To this end, they come up with four paradigms and then show how they are reflected in current protocols. The proposed paradigms are (i) design local interactions that achieve global properties, (ii) exploit implicit coordination, (iii) minimize the maintained state, and (iv) design protocols that adapt to changes.

Accordingly, (Samaan e Karmouch, 2009) presents a more holistic view of the literature in the Autonomic Network management area. Thus, they analyze the requirements and the main contributions for building any autonomic network management system (ANMS). Moreover, they describe a coherent classification methodology to compare existing ANMS architectures and highlight some open challenges and describe new research opportunities.

3.4 Self-* & Normative models

A significant amount of work have dedicated efforts to self-organization and norm-based models principles and characteristics separately. However, to the best of our knowledge, few studies yield relevant progress in correlating both concepts. We believe that one of the major challenges is the disjunction of research in the self-organizing and normative models areas. In the following paragraphs, we provide the current state of the art carrying out on both perspective: self-organizing and norm- based model. In addition, as a strong motivation, we highlight the work of (Samaan e Karmouch, 2009), which is a research study that emphasize the significance of having more dynamics and programmable self-organizing systems in the field of autonomic network.

In (Serugendo et al., 2004), a review of natural and complex systems that exhibit emergent behavior through self-organization is presented. The authors classified the major self-organizing mechanisms used in natural and software systems to achieve self-organization besides demonstrating examples of emergence and self-organizing applications. They also briefly describe the importance of having self-organization in Network problems. As for a MAS perspective, (Serugendo et al., 2006) defines the concepts of self-organization and provides a state of the art survey about the different classes of self-adaptive mechanisms applied in the multiagent systems domain.

On the other hand, for a Norm-based prospect, Norms in multiagent systems are treated as constraints on behaviour, as goals to be achieved or as obligations (Conte e Castelfranchi, 1995). Thus, the author of (Tuomela, 1995) has categorized norms into four categories: rule norms, social norms, moral norms and prudential norms. He also states that Rule norms are imposed by an authority based on an agreement between the members of a society. Moreover, in (López e Luck, 2003), a general model of norms is proposed. They emphasize aspects that autonomous agents might consider before taking decisions regarding the applied norms, while representing the most common kinds of norms and its effectiveness to design a norm-based systems.

Joining both concepts, (Zambonelli et al., 2011) describes schemes that facilitate components to exhibit self-adaptive behaviors and mechanisms to enable components to express the most suitable adaptation scheme. To this end, they consider normative multiagent systems to collectively regulate the autonomic activities of the modeled components. Furthermore, at individual self-adaptation level, they express normative rules to drive the behavior of ensembles and their adaptation constraints.

4
Context-Aware Self-Organizing Model

The Virtual Network Provisioning model itself involves operations such as instantiating, deleting, monitoring, and migrating virtual network elements (routers and links), and setting resource-allocation parameters, in a totally independent and decentralized way, in order to cope with dynamic changes in service demands and resource availability. Such functions make the proposed solution a suitable model for creating and managing multiple VNs and, as a consequence, for supporting the pluralist approach of the Future Internet (Egi et al., 2007). This solution is capable of creating multiple customized VNs and at the same time it exhibits flexible management and real-time control.

Our context-aware and self-organizing model is based on a distributed algorithm, which embeds an autonomic agent composed of an autonomic control loop inside every virtual and physical node throughout the substrate and virtual network. Such agents are responsible for monitoring the local environment, capturing local information, reasoning about measured data, and cooperating with each other in order to exchange their local knowledge and feedback. Thus, each agent represents an autonomic entity capable of inferring the local and global network state and, therefore, supports the core of the self-adaptive model. This core model triggers adaptation plans depending on the surrounding environment.

The adaptive agent leads to the Virtual Network (VN) emerging as a self-organizing entity. The agent is in charge of handling local behavior to enable proper local control and management of the virtual network, its components and the network flows. Thus, this control and management maintain the efficient use of physical resources for network virtualization. Assuming that the VN has already been provided, the proposed adaptive model relies on the following features: (i) self-adaptive behavior; (ii) resource and context-awareness; (iii) knowledge acquisition/sharing; and (iv) autonomic normative monitoring. As a consequence of these features acting in combination, distributed autonomic decision-making occurs and this triggers different adaptive plans to eliminate VN degradation as well as coping with scarce physical resources. Next, we briefly discuss such inherent features.

Our approach differs from previous works in the way we select potential VNs intrinsic properties of VNs that favor self-organizing behaviors along with describing a model for efficient autonomic management. In the following sections, we present our techniques to incorporate self-organizing, context-awareness and normative self-tuning on VN provisioning.

4.1 Critical Scenarios

The virtual network provisioning model is responsible for the adaptive maintenance of the system. It aims to deal with dynamic changes from the variations in the physical and virtual networks, also related to overloads, mobility, migration and maintenance needs to ensure an efficient use of physical resource and a high level of quality of service. Therefore, its main idea is to maintain the original topology and service levels agreements during the virtual network lifetime, even when facing either network degradation or the execution of adaptive plans for virtual routers and virtual links management.

Accordingly, depending on the Network condition (link health/resource availability), the proposed model anticipates a critical scenario and as a consequence, triggers specific adaptive plans. These adaptive events are triggered to get rid of such undesired scenario and maintain the virtual network running in accordance with predefined requirements. To this end, we first need to specify a set of critical scenarios to be addressed by our model. This set of environment conditions represents a mapping between specific scenarios of either virtual link or resource degradation and their correspondent adaptive plan. Once the critical network conditions are identified adaptive plans are pre-coded inside every management agent, so they are capable of triggering self-adaptive events to face these specified environment variations.

The critical scenarios are as follow:

Virtual Router Overload

The replacement of virtual router is an adaptive plan that is triggered when a virtual router suffers from anomalies and failures such as lack of resource (high CPU, Memory RAM, etc. usage), link overload (high bandwidth usage) or whenever it gets unresponsive. This particular environment condition is described as an overcome of some specified threshold, based on network metrics related to link/CPU/Memory RAM usage. The model has a set of thresholds that determines whether a virtual router is overload. We highlight that the main goal of the proposed model is to prevent such events by anticipating them, as a virtual router overload would affect the quality of service, the network stability and, therefore, the main model goal.

Note that, if an autonomic management solution is not provided, the virtual network crash in the case of virtual router overload. The overload caused by lack of resources such as memory RAM or CPU leads to an unresponsive behavior that, therefore, breaks the virtual network. This same consequence happens for the following critical scenarios.

Unbalanced Virtual Link Usage

When the virtual network is attending to different users request, i.e., having multiple package flows coming and arriving from/at different virtual routers, the link usage can present an unbalanced state. Hence, the scenario of unbalanced link is characterized by having packages passing through the same link when there are different alternatives. Having multiple services passing through the same path might lead to a link overload, which is an undesired scenario.

This way, if a virtual router is holding different services, its agent has to first analyze all the possibilities to balance the network load at this point. Thus, this particular scenario might trigger distinct adaptive plans depending on the environment possibilities:

1. Virtual Links reassigning; in the case the virtual network has an virtual router available to receive the link reassignment
2. Creating Virtual router to reassign virtual links; otherwise.

Physical Router Overload

During the lifetime of the VN, because of the dynamism of the surrounding environment, a physical router, P_i , may host multiple virtual routers, V_{ij}. Pa_i and Va_{ij} are the agents assigned to the physical router and its virtual router respectively. As we are not able to control such an arrangement of the resultant VN, when a physical node, P_i for instance, becomes overloaded or exhibits a poor quality of service (affected node), its agent Pa_i, the agents from the neighborhood (cluster), and the virtual agents, Va_{ij}, of virtual routers hosted in the physical router P_i, are able to detect such a critical event. In this sense, a physical router overload is given whenever the physical machine, which may host one or mode virtual routers, gets overloaded. By overload, in this particular scenario, we mean high usage of real resource such as CPU, Memory RAM, etc.

The main issue in dealing with physical router overload relies on the fact that the physical routers share real resources among multiple virtual routers. Thus, in the case of physical router overload, we have to enable the live migration of virtual routers among different physical machines.

4.2 Enabling Self-Adaptation

Network management is autonomic when the network system can make decisions on its own using high-level policies. A VNPS is considered to be autonomic when it exhibits Self-* proprieties such as self-organizing, self-awareness and self-configurability. Thus, it is necessary to identify some intrinsic characteristics of the system that favor self-adaptive behavior, such as means of communication and degree of autonomy. After such properties are identified, the VNPS is designed by applying self-adaptive mechanisms and paradigms.

In this section, we address the following key questions related to a networks self-adaptive properties:

1. What principles do self-adaptive systems and autonomic VN provisioning have in common?
2. What are the design paradigms needed to build a generic self-adaptive system that can be applied in VNMS?

For complex network management, we need to design rules and apply models, bearing these questions in mind so as to facilitate interactions between virtual and physical devices. The first question forms the basis of most self-organizing systems, in that we must distribute the responsibility of management among all the individual devices spread over the network such that each agent contributes to collective emergent behavior, instead of having a single entity in charge of the overall organization. To this end, we have designed local rules and assigned local properties that, in conjunction, automatically lead to the emergence of a global goal: network stability. Therefore, we have reduced the global goal to corresponding distributed local goals, in which entities have their own goals as well as interacting locally with their neighbors to build local views of the environment and make small local adaptation actions.

Another advantage to such an approach is that by distributing responsibilities among entities spread over the network we enable local monitoring and local organization, which leads, as a consequence, to local consequences. This distribution certainly made our model more stable and robust with respect to changes in the environment. Local changes represent local consequences at single points of the network and local failures are not reflected in the whole system and can be handled locally. An application of this concept in our model is illustrated in the rules (Norms) applied to each autonomic agent. Examples of norms are:

1. Maintain the usage of the link at most at a determined threshold
2. Maintain a balanced link usage
3. Keep local knowledge current

The second property that we highlight, which relates our problem to self-organization (Prehofer e Bettstetter, 2005), is the exploitation of implicit coordination. In explicit coordination, direct message exchanges are used to coordinate resources, which is a typical characteristic of centralized systems. In contrast, our model uses implicit coordination, in which self-organization event, context and global status are inferred from the observation of local environment and from relations with neighbors. In this case, each autonomic agent observes the neighborhood and based on such observations, draws conclusions about the state of the network and reacts accordingly. To implement this concept, our model provides a message exchange schema only within the cluster to which it belongs. Specifically this means that an agent only exchanges messages in its own neighborhood. All knowledge beyond the cluster is acquired by inferring from such locally exchanged messages or from further messages exchanged through different clusters.

An example where we applied implicit coordination is in the detection of adaptation event functionality. Basically, each agent frequently exchanges messages with its neighbors in order to update its local knowledge. However, if a physical or virtual router decides to enter into a self-adaptation state, it stops exchanging update request messages in order to save resources. As each agent expects at least a keep-alive message from all its neighbors, if a message is missed the neighbors conclude the router in question might be involved in an adaptive event. Neighbors first check whether the router is up and running, by checking if the link is up; then they infer whether the router is involved in an adaptation task or is no longer responding.

The third self-* property applied to our autonomic model is related to obtaining and keeping local network state information. The author of (Prehofer e Bettstetter, 2005) believes that to achieve a higher level of self-organization, it is necessary to minimize the amount of long-lived state information, in which long-lived state means any necessary information about the network state, either global or local. At run time, the VN and its devices (virtual and physical) need to store information about the network itself. For instance, each autonomic agent needs to store information about the physical and virtual topology, as well as the current IP tables and its direct neighbors. We applied the concept of localized interactions, and implicit coordination and as a consequence we also promoted less state information maintenance. By uniting these approaches, we can take advantage of minimizing long-lived state information. We primarily use the knowledge discovery mechanism concept, in which each agent is able to exchange messages to request an update. If a requested piece of information is not available at the neighborhood level, the request is then recursively passed on to subsequent neighbors.

Finally, as the fourth property, we present the autonomic monitoring and analyzing. Virtual Network monitoring is concerned with the collecting and analyzing of the specific measurements to determine the underlying virtual and physical network status, as well as the possibility of network critical scenarios. In general, traditional monitoring mechanisms rely mainly on the systematic collection of measurements of predetermined parameters collected by the autonomic control loop. Differently from this approach, our model holds an autonomic monitoring mechanism to continuously adjust their operations, parameters and constraint in order to reach an equilibrium between the need of accurate view of the network status and the processing overhead. To this end, autonomy implies the ability to adjust, at runtime, which network components to be monitored, at which level of detail, how to adjust monitoring parameters, when to execute monitoring functionalities and for how long.

4.3 Autonomic Control Loop

The ability of the autonomic agents to react to changes in the network is provided by the adaptation of the Autonomic Control Loop concept, defined by IBM for autonomic computing (Computing et al., 2006) and already discussed on Section 2.3. This concept is comprised of an autonomic manager responsible for managing one or more elements. As our solution is fully decentralized, and there are no centralized entities to handle notification of environmental changes, each autonomic agent has to monitor and analyze its local environment continuously and act based on local variations. To achieve a high degree of autonomy, the core of our self-organizing model supports five main functions: (1) monitor the physical/virtual router; (2) analyze the router performance; (3) plan and make decisions; (4) check norms, to verify that they are applied; and (5) execute a set of appropriate adaptive plans. Such tasks comprise a machine state, where distinct transitions exist between the functions depending on the state of the autonomic agent. Together with a knowledge base, which maintains the necessary information about the entities, its operations and the environment, these functions are referred to as a control loop.

4.3.1 Monitoring

The monitoring function relates to the collection of information including supervising, monitoring and storing the necessary measurements from network links and the physical and virtual resources that are of significance to the self-properties of the underlying network. Because of characteristics enabled in our model, the use of a systematic collection of predetermined parameters has been avoided. Instead, we applied a dynamic approach that relies on continuous adjustment of such operations in order to balance the need for monitoring the view of the network state and the related overhead. To this end, we enable the ability to decide, during the lifetime of the VN, which network components shall be monitored, how to tune monitor parameters, how often to execute monitoring tasks and how long to collect such data.

The monitoring function can be characterized as dynamic since it acts differently depending on the network state; i.e., it receives measurement data from a list of collectors residing in the virtual/physical router, as well as obtaining measurements by directly sending explicit messages to its neighbors. The periodicity of the each monitors run and how long they execute their monitoring tasks are also dynamically determined by the network condition. This approach decreases additional traffic overhead when the network has high resource usage, and enables deeper monitoring when a router has available resources. In this sense, the monitoring tasks are self-tuned and depend essentially on the state of the network in every cluster. We describe how we treat self-tuning in the following topics.

4.3.2 Analyzing

Another component of the control loop is autonomic analysis. It translates the acquired data into local knowledge in order to determine the description of the performance of the underlying network and to check whether the network state is in agreement with the quality of service and required policies. In addition, it also anticipates future critical scenarios and detects events, such as either virtual link overload or physical resource scarcity. In other words, autonomic analysis is the key to activating decision-making in case adaptation is required. In this research, the analysis relies on a set of specific concepts: (i) History-based prediction, in which we define a time window to take into account the history of each autonomic agent rather than only its current state; and (ii) online anticipation. This latter concept is related to the fact that our environment presents a highly dynamic behavior, in which it is not expected to exhibit periodicity over the VN life-time. Thus, we need to provide a way of continuously predicting system behavior using up-to-date knowledge.

In order to maintain efficient use of the physical resources, in which the VN provisioning maximizes the balanced distribution of physical resources among virtual devices, we have introduced a metric to categorize link usage. This metric on resource usage can differ depending on how the resource has been requested. For instance, a link with high and stable usage might trigger a different scenario of a link with medium usage but with increasingly more requests.

4.3.3 Decision-Making

When it is anticipated that critical scenarios may occur, the decision-making function plans for and might trigger the execution of system solutions. Such solutions refer to adaptive operations to re-configure the virtual network topology as well as the balancing of physical resource usage among virtual devices. The core of our self-organizing model makes decisions based on the knowledge retrieved by the Monitor and computed by the Analyzer, as well as from the knowledge exchanged between neighboring nodes. Such decisions depend essentially on the virtual network state, the local knowledge and the prescribed norms, and are based on the choice of previously designed adaptation plans. These plans could include: (i) activating the creation or the deletion of a virtual node; (ii) fine-tuning the amount of virtual resources allocated to a specific virtual node; (iii) migrating a virtual node to a different physical node; and (iv) balancing virtual links.

The key concept behind the decision-making function is that it enables self-organization of its own resources according to the variation of both the substrate (physical) or virtual network. The self-organization occurs through an examination of internal knowledge to decide when, where and how to perform an adaptive plan. To achieve such a degree of autonomy and execute different adaptive plans upon different network scenarios, decision-making functions continuously monitor, analyze and fine-tune the physical/virtual components and lead the system into a more stable and reliable network.

The decision-making function was designed to trigger adaptive plans in response to either external or internal events. The former represents, conditions such as virtual router overload as well as link degradation, while the latter, is related to events in the substrate node, such as lack of physical resources. Such mechanisms are the reverse of what we see when we compare it with traditional approaches where the goals of planned adaptations are related to enhancing the performance of the system, typically for non-critical scenarios.

4.3.4 Norm Checking

The last function of the control loop beside the executor itself is the checking of pre-specified norms. Norm checking refers to the task of verifying whether the conditions of the network and the virtual and physical routers match the set of norms designed for the system. Such norm sets are related to the ability to provide the virtual devices with controlled autonomy, by restricting their behavior to prevent malfunctions and undesirable behavior as well as a way to maintain a dynamic control loop. In this case, the control loop components and behavioral parameters can be tuned, at run time, over the VN's lifetime. Refer to Section 6 for further details on Norms and how they are checked.

5
Towards Context-Awareness

In order to build a virtual Network provisioning model that satisfies self-organizing paradigms, we need to distribute the responsibility of the virtual network provisioning among all physical and virtual devices spread all over the network instead of having a single central agent responsible for the overall organization. Such self-* concepts refer to local monitoring, local reasoning, cooperation among neighbors while making and executing decisions that lead to local consequences. Indeed, the design of distributed rules with local interactions in conjunction with these self-* properties lead to the emergence of the global goal, which is to maintain the virtual network running in accordance with hight quality of services and users requirements. Bearing in mind such a distribution of responsibilities, it turns out that the agent's knowledge process and communication means are the main keys to ensuring that this distribution will handle updated knowledge across all agents. Another advantage the model takes by having such distributed architecture is the efficiency in terms of message exchange to request and inform actions and current status.

As the proposed model presents an agent hosted in every physical and virtual device, when a re-scale of the virtual network is needed, the amount of management agents will also increase linearly with the number of network devices. If we consider a central approach, that handles all knowledge and communication among the virtual and physical devices, it leads to a considerable increase of overhead because of the number of messages exchanged needed. Because of the intrinsic characterizes of the network setup (routed network), the monitoring entity, in a centralized approach, needs to access every virtual and physical router to monitor and analyze their status according to metrics related to resource and links usage. Therefore, the centralized approach seems to be unlikely to work in this specific domain, since, in high usage scenarios, where there is already an overload of links at some points of the network, this will worsen even more the bandwidth usage and the overhead associated.

In order to avoid the use of explicit message exchange and to cope with this inefficiency in the agents' communication, the model's coordination information is done through the implicit exchanging of messages. Thus, the local network status, the event triggering, the adaptive plans requests are represented as knowledge inferred from the local environment observations and relations among neighbors of the same cluster. To achieve such an implicit coordination, each autonomic agent observes the neighborhood and listen to keep-alive messages and, based on local observations, draws conclusions about the state of the network to react accordingly.

To meet the implicit coordination requirements while satisfying the self-* properties, the model provides a communication scheme only within the cluster to which it belongs. All knowledge beyond the cluster is acquired by inferring from any common agent that belongs to a different neighborhood. Moreover, the self-organizing model provides a dynamic autonomic control loop, which dynamically collects information regarding the network status. The use of self-awareness shows up as an important feature to enable autonomy and self-* behavior.

5.1 Agent Communication

The means of communication of the proposed Self-Organizing Model consists an important feature that supports the exhibition of better self-organizing behaviors and actions. It supports describing a proper self-organizing model for the context of autonomic virtual network management, mainly because of its high distribution and localized view. We believe that enabling a localized communication promotes other self-* characteristics, which leads to local management, local collaboration and local consequences. These properties together favor the robustness and scalability of the model. The local property explains the robustness and scalability due to some intrinsic characteristics of the domain in which the autonomic monitoring process should ensure minimum extra overhead while maintaining the updated network status. Indeed, the local property, in which only agents from the same neighborhood (cluster) are allowed to exchange information, leads to a model that has local management, local adaptation that in conjunction leads to local consequences, which explains the robustness.

Some research studies regarding autonomic networks and management/mapping of virtual networks have a semi-distributed model, in which the agents are embedded on physical resources, rather than on physical and virtual routers. Thus, the agent that controls the physical resource is also responsible for controlling all virtual nodes that it hosts. On one hand, considering a semi-distributed scenario, the agent concentrate different knowledge about the network at this location, as it controls both the physical and the virtual activities. This fact is what characterizes the semi-centralized model, in which each agent manage its own physical resources, links usage and also manage every virtual router that it may embed, and, therefore, their management activities. On the other hand, however, this same agent has to access each one of the virtual routers it has deployed, in order to run basic management operations. This may cause a overhead while processing each one of the virtual routers deployed in the virtual network.

It is important to note that unlike the choice of the communication methodology itself, the use of the multiagents to represent this model is essential since it makes use of the autonomic communication among agents of different substrate nodes to gain advantages over traditional approaches to manage virtual networks.

The first goal of this research is to distribute the VN and physical resources' control and management across the substrate nodes. Thus, in order to reach a model capable of self-adaptation, each virtual and physical router has a dedicated embedded agent responsible for the local monitoring and control. The self-* properties, for instance, lead to local consequences, and, therefore, to a solid communication means, as any self-organizing system (Prehofer e Bettstetter, 2005) (Serugendo et al., 2004) (Serugendo et al., 2006).

Maintaining updated information about the environment in a centralized way do not scale well. It also exhibits high overhead and delays in the decision-making process. This happens especially when the underlying domain is highly dynamic and happens to present high variance. In our model, each agent processes and exchanges knowledge through messages and cooperate, in a structured way, to endure decentralized decision-making for the VN. Hence, the proposed autonomic monitoring approach relies on a local view. This local view consists in agent-based local observations of the environment obtained through a passive observation of the internal relations among neighbors and local state retrieved from the knowledge acquiring process.

5.2 From Monitoring to Decision Making

Collecting and maintaining a global view of the network is expensive in terms of resource usage. As a result, our monitoring approach relies on local view, in which each agent makes local observations of its own environment and draws conclusions accordingly. However, to enable a holistic view of the network, in which the agent infers the network-wide view (global view), the agent monitors numerous samples of local views from various agents within the network (Winter e Schiller, 2005). From the keep-alive messages received from the neighbors, the agents build an estimated global view to evaluating its own relative status (Winter et al., 2006). Such monitored knowledge is used for the decision-making process, to trigger proper adaptive plans related to network load balancing, and self-management properties.

An example in which the knowledge process supports a better triggering of specific adaptive plans is when an agent decides to balance the virtual link usage, in the case of link overload. Once the agent anticipates a link overload, it can re-organize its own links in order to rebalance the links usage. Through the awareness of the global state of the network and of the individual routers statuses, the agent is able to steer the selection of virtual candidates based on concrete realization.

5.3 Knowledge Process

The authors of (Samaan e Karmouch, 2009) stated that to achieve an autonomic management thought a precise model of management system, the key concept relies on a built-in reasoning mechanism that leads to knowledge process to acquire and share the knowledge. We also believe that having an autonomic process to acquire and share knowledge lead to an architecture independent on central entities with global information. Bearing in mind that the proposed solution has local reasoning mechanisms which support the decision making to trigger adaptive plans that involve agents from the same neighborhood, there is no need to have a central entity responsible for holding all knowledge of the network. Thus, our model relies on local reasoning to lead to local consequences.

Thus, we enabled self-awareness through:

1. knowledge representation, which can be characterized as behavioral, structural and adaptive, all represented by ontologies;
2. knowledge acquiring/sharing, in accordance with self-organizing paradigms, in which each entity is able to infer local conditions through observations and message exchanges in the neighborhood.

We highlight that limited resources is another property inherited from the domain. As the problem is characterized as highly dynamic and prone to high variations, where frequent local overloads can occur, at any point of the network, the solution is to come up with a model to self-adapt itself and its resources in order to get rid of such an overload scenario. In this sense, the agent's communication has to ensure minimum extra overhead.

Following this belief and aiming to enable cooperation among neighbors in terms of knowledge acquiring/sharing, we propose the use of context-awareness, knowledge discovery and knowledge sharing process in an attempt to increase our system's degree of intelligence and enrich the self-organization model. Like (Samaan e Karmouch, 2009), we also believe that the ability the autonomic entities have to infer the local and global state certainly increases the effectiveness of the decision-making process, as it also is highly related to some paradigms applied to our model, such as minimizing state information and implicit coordination.

To enable the capability of autonomic adaptation, three types of knowledge have been mapped: Structural, behavioral and Adaptive plans. This knowledge is used by the group of agents to collaborate in order to make an individual decision or collectively support an adaptive plan. The autonomic adaptations are related to: (i) creating new router, (ii) migrating an affect virtual router from one host to another, (iii) adjusting setup configuration of the physical and virtual devices, and (iv) facilitating the communication regarding decision making.

The types of knowledge representation are described in the following sections.

5.3.1 Structural knowledge

The Structural Knowledge refers to the domain knowledge, which provides a view and conceptualization of the virtual and physical domains. It represents information about the devices of the modeled domain, such as virtual routers, physical routers, and links, their properties, their relations, and additional information about the architecture of the domain. Hence, technical description of how a virtual machine may be set up is an example of structure knowledge. It also involves technical information regarding Memory RAM, CPU, Bandwidth set up, etc. The physical and Virtual topologies are also examples of structural knowledge. Refer to **REVIEW**: table 1 for further details on Structural Knowledge.

5.3.2 Behavioral knowledge

The Behavioral Knowledge, in turn, represents all different behaviors of the modeled domain, not only the ones that lead to a self-adaptation. It is also considered a domain knowledge and describes all different behaviors the model may exhibit, their properties, the environment condition in which they may occur and their relations. Moreover, it describes the mapping between the environment condition and variations in terms of Memory RAM, CPU, Link usage, package delay or package loss and all possible behaviors that may be addressed to each specific scenario.

We can exemplify a behavioral knowledge through the full description of each component of the self-organizing model, such as the Analyzing function, which is responsible for translating brute data into structured information. This description involves the behavior properties, what data types it analyzes, how long it behave, how often it is executed, and some dynamic possible parameters to manage its own setup. It also describes all components of the network, such as Monitoring, Decision-Making, etc. There is also exist the correlations among different component of the model.

5.3.3 Adaptive knowledge

On the other hand, Adaptive Knowledge is used to describe all adaptive plans. It represents the different ways to manage and control the modeled system in the case of environment change. It includes, for instance, a set of domain related variations and their corresponding applied adaptive solutions. Adaptive Knowledge differs from Behavioral Knowledge in the sense that now it describes, beyond the conditions in which each adaptive plan occurs, the consequences of such adaptation and the expected final environment condition. It also describes how an agent must behave when it faces an execution of an adaptive plan in the neighborhood, like decreasing the number of request sent to the affected node, for instance.

Thus, from an autonomic virtual network perspective, self-awareness is easier achieved when we can express the different components of the modeled system **REVIEW**: though – through a knowledge base. Refer to Table 5.1 for further details on knowledge representation.

5.4 Knowledge Acquiring

With the aim of decreasing the dependency of pre-embedding knowledge and the constant need for synchronization among distributed agents, the model uses the concept of implicit coordination rather than having an explicit knowledge base. As already discussed, the main characteristic of the proposed self-organizing model is the local activity, which leads to local consequences. Having an implicit coordination to gather and share information is also an approach to enriching the properties already highlight in this thesis.

Structural	Behavioral	Adaptational
Physical/Virtual Topology	Behavior of components (e.g., Monitoring, Analyzing)	Critical scenarios description
Possibles CPU size	Bandwidth variation (e.g., metrics)	Environment conditions that leads to adaptation
Possibles Memory RAM size	Execution time of Components	Management actions
Possibles Link Bandwidth	Interval between Components execution	Adjusting agent configuration
Current configuration	Adaptation plans in response to environment variations	
Neighbors addresses		

Table 5.1: Knowledge Representation

Inferring current virtual and physical network topology

A classical example of knowledge inferring, also present in (Prehofer e Bettstetter, 2005) (Kanellopoulos, 2011), is the discovering how the network topology is provided. In this project, we assume that both the physical network and virtual networks are provided when the self-organizing model starts running inside every device. Note that research studies in the Network filed, as already discussed in Section 3, provide solutions for the problem known as Virtual Network Mapping, which provides a virtual network upon a VN request.

Thus, once both physical and virtual networks are provided, each router is built with the description of its own resources, its network setup, and a list of its direct neighbors. However, during the life-time of the virtual network, if an agent had to come up with a decision regarding any adaptive plan, we believe that, knowing the virtual topology will support the agent to make a proper decision regarding the self-organization. For instance, in the case of Live Virtual Router Migration, in which a virtual router is migrated from one affected node to one node that is nearby and available, it is crucial to know the link length between the affected node and any possible destination candidate.

Hence, a candidate that is two links path away from the affected node is preferable rather than one that is three links path away. Therefore, during the VN life-time, the model exhibits a need of having further knowledge about the structure of the entire environment and a global view of the environment itself.

Accordingly, once the virtual network is provided, an adaptive algorithm runs inside every physical and virtual device, in order to infer the physical and virtual topology. To this end, an agent sends a message to its neighbors requesting the topology information - i.e., the neighbors of its neighbors. Each agent that received this request sends further messages to their respective neighbors. Next, once they get the responses, they send a message back to the first agent informing about their neighbors. At the final stage, when the original message is responded to the very first agent that requested the topology information, all nodes from the network have inferred how the topology of the current network is provided.

As we are proposing a model for management purpose, we deal with the high variance of the environment to enable autonomic self-adaptation. However, we have not addressed to changes in the topology yet. Thus, once the virtual network is provided, no node can be deleted from the network. Only insertion of a virtual router can be performed, which does not affect the current topology, as no link will be deleted from the topology. The topology discovery happens in both ways, in the virtual and physical environment.

Inferring adaptive plan execution

Similar to the inferring network topology, inferring adaptive plan is a way to avoid extra message exchanges during the execution of an adaptive plan. Thus, the execution of an adaptive plan is also inferred through the exchange of message or the absence of it. When a local adaptive plan is occurring at some point of the network, the model's main goal is to have only local consequences. **REVIEW**: Therefore As the adaptative actions do not require the involvement of any other agent besides the ones from the same neighborhood, no direct message is require to inform them that an adaptation is occurring. So, the approach of letting the other agents know that an adaptation is being triggered through inference is a desired behavior. Note that, during the adaptation event, the agents of the affected cluster are committed with the accomplishment of this particular adaptation plan.

For instance, in the case of replacement of a virtual router, which is triggered from an overload scenario, the affected node does not require any support from the neighborhood. However, as there is frequently keep alive message exchanged among all nodes of the same neighborhood, whenever a router faces a replacement adaptive plan, it sends an alert message to all nodes of its cluster. Being aware of such an event, these nodes stop sending any message, request or inform message to the affected node until further informative message from the affected agent. Once the adaptation is achieved, the affected not returns sending keep-alive messages with the description of its status after the adaptive plan.

Similarly, the event responsible for balancing virtual links follows the same idea of inferring context. When an agent detects an unbalanced usage of the virtual links, it triggers adaptive plan without any explicit support of the adjacent agents. Once it creates a new router or simple re-assigns an existent link to another virtual router, it sends an update message to all agents from its neighborhood. Next, once these messages are received, each agent is in charge of updating its own routing table, in the case of need.

The third event that might happen during the life-time of the virtual network is related to the live migration of virtual routers. This is the most complex behavior that our model might exhibit. Since it is an adaptive plan that occurs because of an overload of the physical machine, it requires the support of all nodes of the affected node cluster. In this case, it is needed to minimize the message exchange among the agents that participates of the adaptive plan and interrupt, for a certain amount of time, the extra messages exchange between two different clusters. As the model is a proposed solution to minimize the impact of links overload, the unnecessary message exchange worsens the overload scenario. Bearing this in mind, when an agent gets involved with this adaptive plan, it stops sending keep alive message to its neighbors from the other clusters, so that the other agents are able to infer the occurring of this adaptive plan from the absence of the keep-alive messages.

Implicit coordination

Implict coordination is the base of the model's knowledge acquiring. As stated before, we distributed the management responsibility among agents spread all over the physical and virtual devices to ensure the local property exhibition. Thus, having a network domain, on which characterizing an overload scenario is still an unpredicted event, even in the Network field, forces the proposed solution rely on minimizing state information and explicit/directly message exchange. The local behaviors and consequences in conjunction with self-* concepts facilitate the use of implicit coordination. Thus, depending on the environment condition, only keep alive messages are allowed in the model. Such messages carriers, besides the information of being alive in the environment, the local status of each router and some extra information, depending on the current situation. From all the messages received during its life-time, a node can draw conclusions about its own environment and also about the global status of the network. To this end, the network and routers statuses are modeled as behavioral knowledge (metric-based), and are stored in a timeline fashion inside of the knowledge base of every agent.

6
Towards Normative Self-tuning

A self-organizing system is defined as a system that can adjust its own behavior and function in real-time without any external support in response to its perceptions of the environment it inhabits. To this end, a self-organizing system continuously monitors environmental variation and reacts accordingly. Hence, an abstract self-organizing model has a set of characteristics to be monitored and a set of pre-defined adaptive actions. These actions represent possible constrained adaptations to overcome any critical condition perceived in the environment.

The major advantage of a self-organizing system is its ability to evolve and enhance continuously the applied adaptation strategies, by learning from previous experiences. Thus, to enrich the proposed self-organizing model we aim, throughout this section, show how to enable self-tuning in the core of the self-organizing model, through the concept of norms.

6.1
Approaching Self-tuning

The self-tuning concept is a feature of norms and how they are used by the model in order to enrich the autonomic adaptation of the managed components. Thus, self-tuning refers to the model's ability to modify a set of its own parameters, thresholds and constraints dynamically at run-time during its lifetime. This modification supports minor adaptation operations.

The main principle behind self-tuning is the capability of enabling minor autonomic adaptations within the core of the self-organizing model, so that the model can learn and adapt itself when it perceives new environmental conditions and needs. For instance, it can dynamically adapt the control loop's running rate, or even monitor the environment for longer than the default period. Moreover, it is in charge of executing small changes in local parameters to discover a better setup for the autonomic agent with respect to the state of the local environment.

Furthermore, the norm concept applied to self-organization provides managed entities with a controlled autonomy. It can restrict the activities of the model's components while preventing malfunctions and undesirable behavior. Indeed, by enabling controlled autonomic events and internal adaptive behaviors, the model also gains a higher degree of autonomy, dynamism and programmability, since it fine-tunes its local parameters, thresholds and constraints according to current local environmental conditions (Localized control and management are intrinsic in self-organizing models). Thus, we believe that norms and self-tuning are together responsible for a more dynamic behavior, more independent of pre-embedded configuration, knowledge and setup. The core of the self-organizing model, i.e., the autonomic control loop, is also able to self-adapt, by fine tuning some of its local parameters to adapt to environmental needs better.

In order to enable a normative behavior in self-organizing models, we propose the use of an extension of the autonomic control loop, MAPE-K, discussed earlier, by adding a new component, the norm checker. This new component is responsible for controlling the different sets of norms, depending on the context in time and space, regulating when and how a specific norm is valid, enforcing sanctions and rewards, in case of violations and good behavior respectively. More precisely, the norm checker manages how a norm evolves from experience and interactions with the local environment. Thus, the norm checker component is responsible for a higher degree of autonomy and programmability, since it is now able to fine-tune itself depending on experiences with the environment.

6.2 Exteding the Autonomic Control Loop

As stated earlier, the autonomic control loop is composed of a norm checker component as well as components for monitoring, analyzing, planning, and execution. The norm checker is responsible for checking a set of predetermined norms, depending on the context, and applying their respective actions whenever they are accepted. In this sense, the autonomic control loop itself is also in charge of controlling the acceptance and refusal of a specific norm, besides its own default behaviors, such as monitoring and analyzing, as depicted in Figure 6.1.

Ideally, once a set of pre-determined norms is captured via the norm checker component, the norms should be directly executed by the norm checker. This is the way an agent interacts with the local environment and the autonomic control loop, and dynamically fine-tunes itself. Thus, we say that a set of norms are applicable when the current state of the environment, internal and external (i.e., inside the autonomic control loop and in the local agent

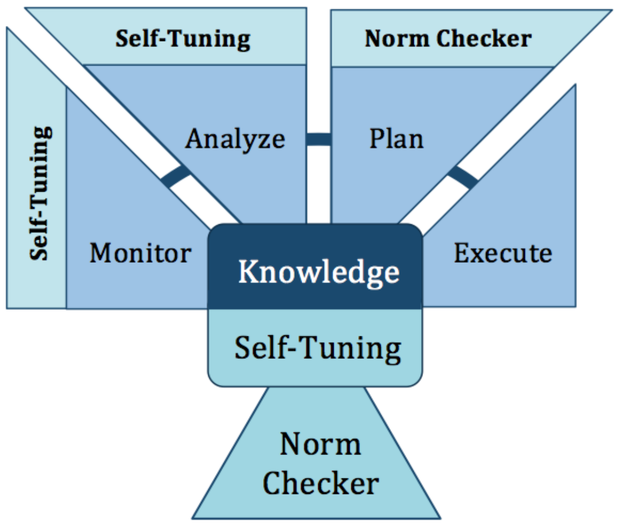


Figure 6.1: Normative Autonomic Control Loop

environment) complies with a set of conditions. The set of pre-determined norms that are checked at every execution of the autonomic control loop depends on the local contextual environment.

We join the concept of norms with the autonomic control loop as a mechanism to govern the self-adaptive behavior of the agents, especially in those cases when an agent is dealing with a self-adaptive plan that might affect the environment and other agents. In this research, the norms are characterized by their prescriptiveness (López e Luck, 2003) , in which each norm describes how an agent must behave in pre-determined situations, which drives the overall behavior of the self-organizing model. Thus, norms specify behavioral patterns for the self-organizing agents, which are represented as actions to be performed in order to change the state of the environment and behavioral restrictions depending on the context of the adaptive agent.

Depending on the context and the domain itself, a different set of norms may be applied, since the norms are applied only in particular circumstances or within a specific situation. Furthermore, the norms are designed to include rewards when they are satisfied by an addressed agent, or punishments when they are not. In this thesis, the rewards are given as an execution of actions followed by an increase of the agent's reputation and fine-tuning of its own parameters. For punishments, a decrease in reputation is given, and its own parameters are fine-tuned. In both circumstances, self-tuning is executed in order to give the autonomic control loop the right balance. Nevertheless, the use of reputation is a way to classify the agents that have a central goal of maintaining a high reputation.

As the VN provisioning solution is fully decentralized, and there is no centralized controller to handle notification of environmental changes, each autonomic agent has to monitor and analyze its own local environment continuously and act according to local variations. Thus, the core of the self-organizing model is tasked, as discussed earlier, with five main functions:

Monitoring

the physical/virtual resources

This function frequently monitors a set of local parameters related to virtual/physical link resource health. In the case of a normative autonomic control loop, the set of monitored parameters may vary depending on the context of each agent. Thus, a set of local parameters is defined as a subset of one or more elements of the set *(link*: *(bandwidth, package loss, package delay, errors, packages out of order, etc.), resource*: *(CPU, memory RAM, IO, etc.))*. This set of local parameters to be monitored depends on the set up of the autonomic control loop, which in turn is defined depending on the acceptance rules of a norm and their respective self-tuning functions.

Hence, in order to avoid the systematic collection of predetermined parameters, the autonomic control loop uses a dynamic approach that relies on continuous adjustment of its own operations in order to balance the need for monitoring the view of the network state and the related overhead. Thus, the running time, the periodicity and the set of parameters to be monitored are examples of parameters that may be fine-tuned by the normative autonomic control loop. Indeed, this dynamic set of parameters responsible for the monitoring function may be fine-tuned, at run time, depending on the environment conditions and on the manner in which the agent fulfills a set of specific norms.

An advantage of using a self-tuning mechanism to control the behavior of monitoring functions relies on the fact that the periodicity on which each monitor runs and how long they execute their monitoring tasks are dynamically determined by network health. Thus, the ACL decreases additional traffic overhead when the network presents high resource usage and enables deeper monitoring when a router has available resources.

Analyzing

performance

This function is responsible for translating the monitored data into local knowledge that can describe the performance of the underlying network at this specific location. In this normative autonomic control loop component, the self-tuning can fine-tune parameters such as (i) the running time, (ii) the analyzer type used to translate data and (iii) the periodicity in which the analyzer perform its actions.

As already stated, an analyzer function contains different analyzer types which are executed depending on the environmental conditions and current needs. As the monitoring tasks, the analyzer type is used to translate data into local knowledge that is dynamically determined depending on network health.

Planning

and making decisions

Decision making anticipates critical scenarios, and might trigger the execution of adaptive operations to re-configure the virtual network topology as well as the balance of physical resource usage among virtual devices. The adaptive actions depend primarily on the virtual network state, the local knowledge and the prescribed norms, and are based on the choice of previously designed adaptation plans.

The main idea behind the normative autonomic control loop is that, through a decision-making function, it enables self-organization of its own resources according to variations in the environment.

In order to control the model's autonomic behaviors and adaptive actions based upon environmental variations, the decision-making process uses a specific set of adaptive norms, which describes the network conditions and the respective actions to be executed in case of need. Thus, we enable a built-in normative system to control all adaptation functions of the normative control loop. An example of such adaptive norms is the triggering of the replacement of a virtual router in a virtual network, depicted in Figure 7.11.

If an agent autonomously decides to fulfill this required norm and does not comply then an adaptive plan is triggered that is responsible for replacing the affected virtual router with a new virtual router capable of handling the current network needs. In order to increase the local knowledge base, the agent saves how often it triggers adaptive plans so as to support decision making in later analyzes. Furthermore, in this norm, resource usage may address such measurements as CPU or Memory usage.

Norm 1: Replace Virtual Router Plan

Norm Goal: Maintain the resource usage limited by an specific threshold

---

Addressees: Virtual Agents
Context: Stable environment condition
Reward: Enable a minor Self tunning, by increasing the monitor and analyzer components
Punishment: Execute the Replacement of virtual router adaptive plan and increase the number of adaptive plan executed

Figure 6.2: Replace Virtual Router Norm

If resource usage has been stable over the life-time of the virtual network then there is compliance with the norm. Compliance means that, self-tuning is executed for minor internal organization such as adjusting the autonomic control loop components.

We highlight that each agent's normative goals and the network's goals are the same. Each agent and the network wish to maintain stable link usage and a high level of quality of service. Thus, this system does not have conflicts between its main goal and the normative goals.

Norm Checking

Checking norms is the responsibility of the default autonomic control loop and also the components that give the proposed model the normative functionality. The loop is in charge of regulating which set of pre-specified norms may be applied as a consequence of the current state of the environment, and also verifying whether the agent is consistent with the norms. Hence, norm checking refers to the task of verifying whether the conditions of the network and the virtual and physical routers match the set of norms designed for the system.

Such norms sets are related to the ability to maintain a dynamic control loop. In this case, the control loop components and behavioral parameters can be tuned, at run time, over the VN's lifetime.

These tasks become a machine state, where there exist distinct transitions between the functions depending on the internal state of the autonomic agent. With the support of the knowledge base, in which each agent stores its own conclusions about the environment and also all translated data from its internal state (collected by the monitors and analyzed by the analyzer), these functions are called a normative autonomic control loop.

6.3 Enabling Self-tuning

Self-tuning is enabled from the correlations among how different sets of norms are applied and fulfilled in the model. Thus, self-tuning mechanisms based on the normative concept aim at achieving controlled external adaptive behaviors and minor internal parameter adaptation. This occurs while exhibiting a high degree of programmability and integration with environmental conditions resulting in indirect interactions between agents and agents and the environment. The main adaptive behavior leads towards the desired global environmental condition, which is a stable network flow.

Thus, self-tuning also occurs whenever an agent disobeys a specific norm, as Norm 2 shows (Fig. 6.8). The agent's goal is to enable fine-tuning of its parameters relevant to the autonomic control loop function in order to self-adapt based on its local environment and current needs. Because of the distributed nature of the environment, in which we have different clusters (neighborhoods), with different environmental conditions and needs, each router, and therefore, its agent, requires a different setup. For instance, considering Figure 6.3, a border agent tends to have a smaller request rate in comparison with a central router. In this case, the red router has connectivity equals to five while the green router has connectivity equals to one, which means the red nodes tend to receive more messages and requests. Since a central controller that sets up and monitors every router on the network, at run-time, suffers from scalability, we aim, through self-tuning to enable autonomic fine-tuning of each agent based on variations in its local environment.

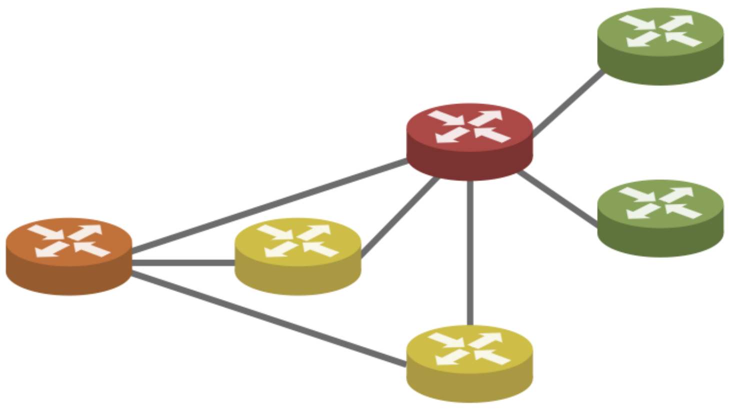


Figure 6.3: Network Topology Impact

[0]Note: This figure shows the impact of having two routers with different connectivities (neighbors). The red router has connectivity equals to five, while the green ones has connectivity equals to one.

The key efficiency concept behind our model is in the self-tuning by triggering adaptive plans. This idea increases the flexibility of the control loop roles in either selecting a better match to support the adaptation task or by analyzing the current network state differently.

To define the formal design by which the organizational norms are specified we show a set of all components that the autonomic control loop can adjust along its life-time. The self-adaptive agent is shown in Figure 6.4, the norms and normative autonomic control loop components are in Figure 6.6 and Figure 6.5, respectively. Finally, the self-tuning parameters are in Figure 6.7.

$$
\begin{array}{l}
\text{\_\_} \mathit{Self - AdaptiveAgent} \text{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
\mathit{NormativeAutonomicControlLoop} \\
\mathit{goals} : \mathbb{P}\ \mathit{Goals} \\
\mathit{knowledge} : \mathbb{P}\ \mathit{Attribute} \\
\mathit{adaptivePlans} : \mathbb{P}\ \mathit{Norms} \\
\hline
\mathit{NormativeAutonomicControlLoop} \neq \emptyset \\
\mathit{goals} \neq \emptyset \\
\mathit{adaptivePlans} \neq \emptyset \\
\hline
\end{array}
$$

Figure 6.4: Self-Adaptive Agent Notation

The self-adaptive agents (Fig. 6.4), *Self-AdaptiveAgent*, contain a set of all pre-determined adaptation actions, *adaptivePlans*, and the normative autonomic control loop, *NormativeAutonomicControlLoop*, depicted in Figure 6.6, as well as a knowledge base, *knowledge*. For sake of simplicity, we highlight that the *Goals*, *Attribute* and *Actions* are given notation. Note that the normative autonomic control loop has a fixed set of tasks which runs frequently. The frequency and the running time of these behaviors are all flexible and tunable through the self-tuning feature. The knowledge base is also flexible in the sense that it enables a window size, in which the normative autonomic control loop (NACL) gathers information from a specific time period to run its own analysis.

A *Norm* (Fig. 6.5) contains a *goal*, a list of *addresses*, that represents the group of agent which may be tied to this norm, a *context*, to represent the condition in which a particular norm is activated and a pair of *reward* and *punishment*, which represent the actions that the agents may take in the case of good or bad behavior, respectively. The set of pre-determined norms are addressed to any *Self-AdaptiveAgent*. In this thesis, a *Self-AdaptiveAgent* can be a *PhysicalAdaptiveAgent* or a *VirtualAdaptiveAgent*.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{Norm}} \\
\hline
goal : \mathbb{P}\, Goal \\
addressees : \mathbb{P}\, Self - AdaptiveAgent \\
context : \mathbb{P}\, Attribute \\
reward, punishment : \mathbb{P}\, Actions \\
\hline
addressees \neq \emptyset \\
context \neq \emptyset \\
reward \cup punishment \neq \emptyset \\
\hline
\end{array}$$

Figure 6.5: Norm Notation

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{NormativeAutonomicControlLoop}} \\
\hline
collector : \mathbb{P}\, Monitor \\
analyzer : \mathbb{P}\, Analyzer \\
planner : \mathbb{P}\, Behavior \\
normChecker : \mathbb{P}\, NormTunner \\
executer : \mathbb{P}\, Context \leftrightarrow AdaptivePlan \\
\hline
Monitors ::= OSMonitor \mid LinkMonitor \mid MetricMonitor \mid DefaultMonitor \\
analyzer ::= DefaultAnalyzer \mid OSAnalyzer \mid MetricAnalyzer \mid LinkAnalyzer \\
NormTunner ::= NormMatcher \mid VirtualAdaptivePlans \mid PhysicalAdaptivePlans \\
executer ::= \\
[\![VirtualRouterOverload \mapsto ReplaceVirtualRouter, \\
VirtualLinkOverload \mapsto BalanceVirtualLink, \\
PhysicalRouterOverload \mapsto LiveMigrateVirtualRouter]\!] \\
\hline
\end{array}$$

Figure 6.6: Normative Autonomic Control Loop Notation

The normative ACL (Fig. 6.6), *NormativeAutonomicControlLoop*, contains a set of possible statuses for each of its own components. For instance, the collector component may contain a list of one or more monitors. Depending on the local environmental conditions and current needs, the normative ACL acts in a different manner. Therefore, the elements that compose each autonomic control loop are dynamic over time and depend on the environment condition. Note that, there are other dynamic parameters that define the ACL besides only the list of components for each of its elements.

Thus, to self-tune each agent of the system, the norm checker component counts on a set of tunable parameters, the *Self-TuningParameters*, as depicted in (Figure 6.7). Moreover, based on the self-tuning function triggered by the fulfilment of a specific norm, different parameters are adjusted. As self-tuning enables a dynamic changing of the tuning of each agent and its respective

$Self - TuningParameters$

$monitor : OSMonitor \mid LinkMonitor \mid MetricMonitor \mid DefaultMonitor$
$analyzer : DefaultAnalyzer \mid OSAnalyzer \mid MetricAnalyzer \mid LinkAnalyzer$
$monitorFrequency, analyzerFrequency : \mathbb{P}\, Attribute$
$monitorRunningtime, analyzerRunningTime : \mathbb{P}\, Attribute$
$analyzerWindowSize, kbWindowSize : \mathbb{P}\, Attribute$
$ACLFrequency, kbVariationRange : \mathbb{P}\, Attribute$
$linkThreshold, osThreshold : \mathbb{P}\, Attribute$
$boundaryMaxFreq, boundaryMaxRuntime :: \mathbb{P}\, Attribute$

Figure 6.7: Self-tuning Parameters Notation

functions, the norm checker contains a list of thresholds that define boundaries for the self-tuning actions to ensure that tuning will lead to an efficient setup. These thresholds, unlikely the ACT parameters, are fixed, defined by an external administrator, and embedded in each agent.

As an illustrative example, *Norm 2* and *Norm 3*, Figure 6.8 and Figure 6.9 respectively, show real norms applied to the domain of the self-organizing model for virtual network management already explained.

Norm 2: Cooperation

Norm Goal: If support is requested, respond whitin a threshold time

Addressees: Virtual Agents; Physical Agents
Context: Pre-adaptive plan environment condition
Reward: Increase the number of request responded
Punishment: Enable a minor Self tuning, by drecreasing the time of management functions: monitor and analyzer components.

Figure 6.8: Support Cooperation Norm

In *Norm 2* (Fig. 6.8), which describes a cooperation norm, the norm *goal* is to maintain a high rate of request response. Thus, the norm states that the execution of an adaptive event must be as fast as possible when responding to a supportive request. In the case of adaptation, an agent may request support from other agents to execute or finish an adaptive action. Because of the complexity of the domain, in a case such as link overload if the agent does not complete a self-adaption on time, the virtual network might crash and stop responding. If an agent takes a long time to respond to a supportive request, which would delay the adaptation itself, it has to trigger a minor self-adaptation (self-tuning) in order to program its autonomic control loop better and respond faster to the next request. The main idea behind this norm is to decrease the overall running time of the control loop, so that it is available to respond to requests in further requests.

Norm 3: Link Boundary

Norm Goal: Keep all virtual links usage rate limited by a specifc threshold

---

Addressees: Virtual Agents;
Context: Pre-adaptive plan environment condition
Reward: Increase the stability metric
Punishment: Increase link stress and executes the self-tuning in order to enabling a deeper Analysis function.

Figure 6.9: Link Boundary Norm

From the virtual network perspective, the link boundary norm (Fig. 6.9), *Norm 3*, describes the threshold of the desired condition for all virtual links of the environment. In the case of link overload, or high link usage, which is not in compliance with this norm, the norm checker enables self-tuning of the analyzer component, letting it execute a deeper analysis of this specific link. As depicted in Figure 6.7, the analyzer is composed of a list of different translators. In this example, self-tuning evaluates the analyzer behavior within the *LinkAnalyzer* and *MetricAnalyzer* components instead of having a default Analyzer.

The model uses different organizations, which can be controlled and dynamically tuned through norms. As the scope of this research is to demonstrate the feasibility of having a self-organizing core with the support of normative systems, we only demonstrate a few examples of the union of these two concepts.

7
Evaluation

Our evaluation focuses primarily on quantifying the effectiveness of our model when applied to a highly dynamic VN environment in respect to overhead and effectiveness in executing adaptations without any external intervention.

We consider the following work in our evaluation: adaptive virtual network maintenance in (Houidi et al., 2010), which presents results by comparing the number of message exchanges while running adaptive events. Hence, we use the following as the main evaluation tools: (i) the comparison among the number of messages needed to cope with stable and critical scenarios; and (ii) comparison of self-organizing model with self-tuning features and self-organizing model with fixed monitoring mechanisms.

7.1
Experimental Setup

In this section, we provide the environment details as well as the initial experimental results to evaluate the efficiency of the proposed model and its adaptive plans.

To the best of our knowledge, few studies in the literature have gone further in this direction, by exploring Self-* capabilities of such VN management. Because of some similarity to our research, we use (Houidi et al., 2010) as a baseline for our performance analysis. As the authors evaluated their solution by measuring the number of message exchanges that may occur in the system, by varying the number of substrate nodes of the network, we also evaluated our proposed model in the same manner. To validate the model's feasibility and scalability, we attempted to test it in a real environment, using a real small physical network composed of five machines. For further details, refer to Appendix B. After observing the behavior of such a model on this real network, we validate it by running simulations for larger networks. We highlight that this project focuses on the appropriate choice of an autonomic plan to manage virtual routers and physical resources efficiently by allowing them to re-organize regularly, rather than only in a migration mode.

To this end, we initially set up a VN infrastructure, as depicted in Figure 7.1, on the top of a physical topology. To take advantage of customized virtual machines with different setups, we have used XEN as a virtualized operating system provider to perform the roles of multiple independent virtual machines on a shared substrate.

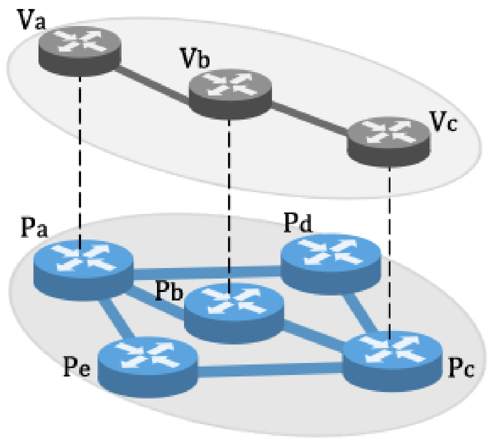


Figure 7.1: Experiment setup

Through these initial experiments, we were able to answer key research questions that serve two purposes namely to: (i) state if the solution is efficient and what contributes to this efficiency; and, (ii) determine if the proposed solution is scalable.

First, we assessed the efficiency of the model, measuring the delay of communication together with the total amount of time incurred to adapt the VN when a virtual or physical device performs poorly or is overloaded. Next, we evaluated the performance of the model by tracking the total number of messages exchanged among embedded agents during the execution of the adaptive plan.

To address the first question, because of a lack of physical equipment to validate our model in a larger network, we simulated a larger virtual environment. The simulation is composed of the same environment setup as the initial experiment, by applying the behavior pattern of the proposed model to get an initial estimate of how the proposed model would behave in larger and more realistic networks. This simulation sought to answer the same question as (i) but for larger environments. We were able to validate all the different modules and aspects of our approach, such as self-organizing paradigms, knowledge sharing and self-tuning, covering every concept of our solution separately. Then, we extended our validation by evaluating the model, considering all features together, simulating a larger network and running the same scenarios once we validated the model, considering the individual components.

7.2 Evaluation Results

An important factor that impact the evaluation of the effectiveness of our model is the cost (in terms of number of messages exchanged) incurred by the network topology complexity and the router's degree of connectivity. In what follows, we present some possibilities of physical and virtual topologies and determine, for each case, the average number of messages needed to carry the network management.

Before analyzing the critical scenarios themselves, we first analyze how the network behaves. We analyze behaviors in terms of messages exchanged and fine tuning operations, when we enable our management solution. Hence, we show that for a stable environment, in which the network loads are balanced over the network links, the number of messages needed to maintain the management tasks running remains constant over time, during the lifetime of the virtual network. We also show how the self-tuning enables a dynamic fine adjustment of the autonomic control loop in a stable environment.

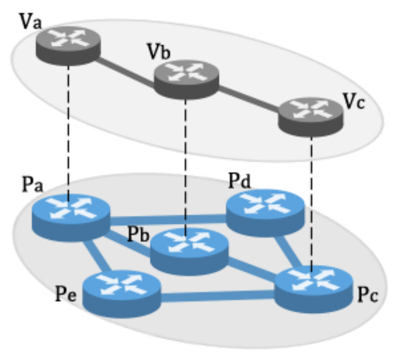
7.2.1 A Stable Network Load Analysis

A message exchange analysis

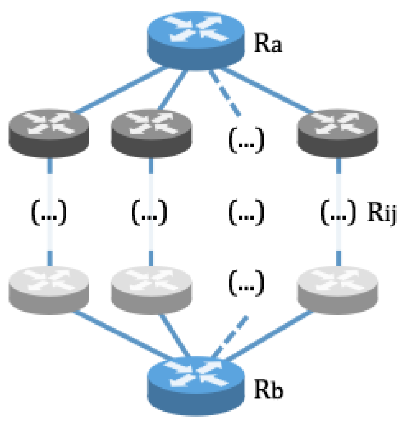
In order to measure the number of messages needed to maintain a virtual network running in accordance with the proposed model, we have set up different virtual networks varying the number of virtual components and the physical and virtual topology.

First, consider the Figure 7.2 and Figure 7.3 to analyze how the model acts, in term of messages exchange, while maintaining the network management mechanism running accordingly. In the case of a stable load over the physical and virtual links, we highlight that the average number of messages needed are heavily dependent on how complex and how connected the networks are.

Figure 7.2 (a) depicts the environment setup for the test-bed experiments while figure 7.2 (b) shows how complex and connected a topology might be. In this figure, we have $\mathrm{R_a}$ and $\mathrm{R_b}$ as border routers, where j means the number of routers between $\mathrm{R_a}$ and $\mathrm{R_b}$ and i the number of distinct paths that connect $\mathrm{R_a}$ and $\mathrm{R_b}$. Of course, there are many topology possibilities, in which each



(a)



(b)

Figure 7.2: Virtual Network setup and possible topologies

router has its own connectivity, but through this picture we aim to clarify how the connectivity impacts our model. Indeed, a dense network carries out a larger amount of messages in comparison with a sparse network. In the case of adaptation, however, what really matters in this analysis is the length of the paths between two border routers - between R_a and R_b, for instance.

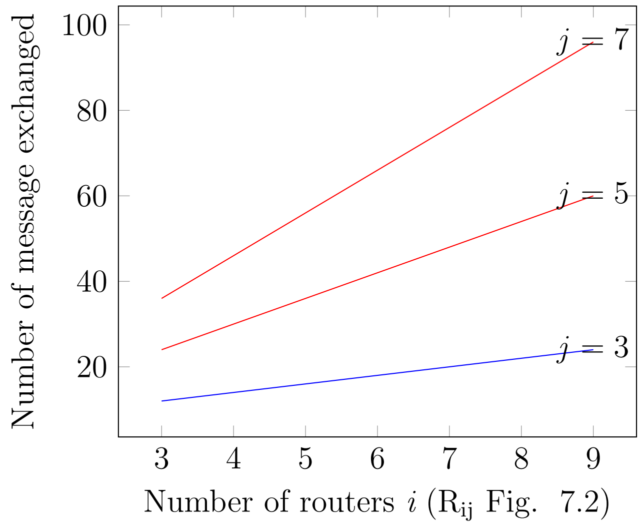


Figure 7.3: Stable VN: Message exchange Analysis

In Figure 7.3, we have plotted the average of messages needed for different sizes and complexity of Virtual Networks considering a stable environment. By stable environment we mean that the model achieves stability regarding network loads and, therefore, presents a stable rate of messages exchanges- that represents the ordinary messages for the maintenance of the VN. In this chart, then, we vary the values of i and j represented in Figure 7.2 (b). As our model relies on the property of locality, the agents only exchange messages on the same cluster they belong. Hence, the number of messages is proportional to the degree of the connectivity of the routers of the network. For example, the virtual network depicted in Figure 7.2 (a) exchange around 4 messages per cycle, while the physical one exchange 12 messages in an average per cycle.

In what follows, we also show that, in the case of adaptation that requires supportive messages exchange to perform the repair action, inside the affected cluster the number of messages needed to execute the respective adaptive plan depends on the number of routers existent between two border routers. By border routers, we mean the first common router of the topology that is able to support the execution of the adaptation. In Figure 7.2 (a), for example, we have P_a and P_c as border routers, and the number of routers between P_a and P_c determines the number of extra messages nedeed for the case of adaptation support. Indeed, depending on the adaptation plan, the border routers support is necessary while executing this particular adaptation event. Bearing this in mind, considering the Figure 7.4, we vary the axis i and j in order to measure the number of messages needed to maintain the virtual environment in the case of live migration of virtual router scenario. Further details regarding this analysis are on the following sections.

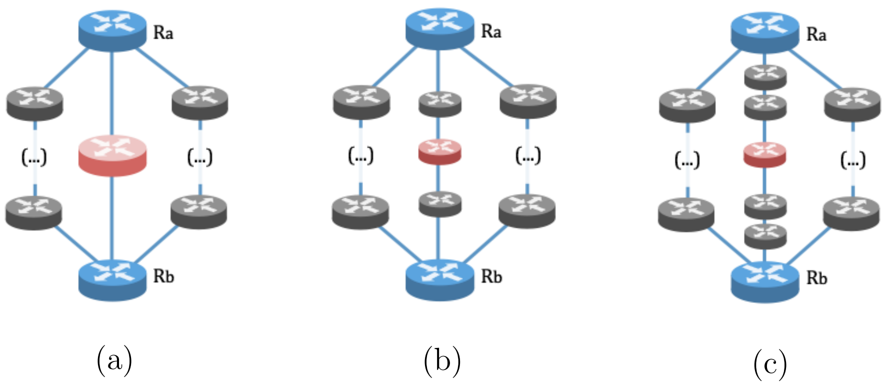


Figure 7.4: Topology impact on Message exchanging

A Self-tuning Analysis

In the case of a stable environment, in which the links and, therefore, the agents spread all over the network do not suffer high variation, the normative autonomic control loop enables a specific set of norms to be fulfilled. Note that these norms for a stable environment are only enabled for those agents whose meet the network stability requirement at their location.

As an illustrated example, as depicted in *Norm 4* (Fig. 7.5), once the virtual network is provided, agents start running their tasks in every virtual and physical device of the network. As they are completely decentralized, and

Norm 4: Functions Rate Boundary

Norm Goal: Keep Functions rate (# messages/ ACL running time) bellow a specific threshold

---

Addressees: Virtual Agents; Physical Agents
Context: Stable environment condition
Reward: Increase the stability metric
Punishment: Executes the self-tuning in order to regulate the ACL running time.

Figure 7.5: Functions Rate Boundary Norm

different points of the network require different behaviors, each agent tends to achieve a different ACL set up. In this case, for instance, a border agent tends to receive less request and messages if we compare with a central agent; thus they present different levels of popularity. In Figure 7.2 (a), we see the virtual routers, V_a and V_b, in which V_a is a central router and V_b a border router. From the moment the environment is provided until they get a balanced setup, each agent, including Va_a and Va_b, constantly fine tune their own parameters in order to fulfill the *Norm 4* (among others), which states to maintain the agent tasks rate bellow a specific threshold. The task rate is defined as a rate between the number of messages received by the executing time of the autonomic control loop. In this scenario, for instance, note that the central router V_a receives six messages per cycle in average while a border routers as V_b, receives three, in average. Thus, depending on the network topology, the request rate presents a high variation regarding borders and centrals routers.

Indeed, the agent Va_a, which is a central agent, tends to receive more messages than the agent Va_b - a border router. Thus, in order enable Va_a and Va_b to prompt respond to other agents requests and taking this disadvantage out, the executing time of their ACL tasks has to be setup according to their availability. Therefore, in this scenario, the Va_a's ACL runs faster if we compare with the ACL tasks of the agent Va_b.

In this experiment we run a stable virtual network, in which every agent has started with the same ACL set up: pre-determined ACL running time an frequency. As the central routers tend to receive more messages and request, the self-tuning keeps running inside every agent in order to balance their function rate. Therefore, in this stable scenario, in which no adaptive events occurs, all agents achieve, after three periods max, a stable and different ACL setup. In the case of adaptation, if such dynamic adaptation is not enabled and as every agent has the same pre-determined set of parameters, a border agent is more likely to not respond to request in time.

Context	Router,period #0	period#1	period#2
Stable	$V_{a,2}$ (1, 1, 2.5)	(1.3, 1.3, 3.0)	(1.3, 1.3, 3.0)
	$V_{b,5}$ (1, 1, 2.5)	(0.7, 0.7, 2.0)	(0.5, 0.5, 1.5)
(no ST)	$V_{a,2}$ (1, 1, 2.5)	(1, 1, 2.5)	(1, 1, 2.5)
	$V_{b,5}$ (1, 1, 2.5)	(1, 1, 2.5)	(1, 1, 2.5)
Adaptation	$V_{a,3}$ (2.5, 2.5, 5.0)	TO, (2.0, 2.0, 4.0)	RD (1.5, 1.5, 3.3)
	$V_{b,3}$ (2.5, 2.5, 5.0)	TO, (2.5, 2.5, 5.0)	TO, (2.5, 2.5, 5.0)

Note: This table shows two different context scenarios, where $V_{a,2}$ means virtual router named *a* with connectivity equals to 2 (number of neighbors) and (Mt, At, Ot) means Monitoring running time, Analyzer running time and ACL Overall running time. TO, in the Adaptation scenario, means time out and RD means the agent responded to a support request.

Table 7.1: Comparison between Self-oganizing model with and without self-tuning (ST) feature.

In Table 7.1, we show how self-tuning affects the model. As already stated, once the virtual network is provided, every router embeds a set of predetermined ACL parameters. If self-tuning is a disabled feature, the agents will remain with the same ACL setup during the virtual network life-time, even though they present different needs. On the other hand, however, if the self-tuning is enabled, each router achieves a different balanced setup, according to their environment conditions. In addition, depending on the initial ACL setup, in the case in which the self-tuning is disabled, some agent would not be able to respond to a support request and neither to fine-tune its tasks, as shown in the time out (TO) event in Table 7.1.

7.2.2 Migration scenario

During the lifetime of the VN, because of the dynamism of the surrounding environment, a physical router, P_i, may host multiple virtual routers, V_{ij}. Pa_i and Va_{ij} are the agents assigned to the physical router and its virtual router respectively. As we are not able to control such an arrangement of the resultant VN, when a physical node, P_i, becomes overloaded or exhibits a poor quality of service (affected node), its agent Pa_i, the agents from the neighborhood (cluster), and the virtual agents, Va_{ij}, of virtual routers hosted in the physical router P_i, are able to detect such a critical event through regular message exchanges. From the moment Pa_i triggers an adaptive plan, adaptation algorithms run inside each agent from the same cluster in order to execute the adaptive plan collaboratively. Only agents from the same cluster are allowed to collaborate in order to determine alternative hosts to which the virtual routers, their agents and their services will be migrated. Basically, each neighbor of the affected router selects pre-candidates, and then the agent of the affected node determines the final candidate host. Algorithm 1 describes how such steps are taken from the moment a physical machine is affected until the adaptation plan is completed.

Algorithm 1 Adaptive VN live migration

```
Precondition: physical machine $P_i$ has detected an overload though its agent $Pa_i$

 1: $P_i$ triggers an adaptation plan: Live migration
 2: $P_i$ sends a pre candidates list request to its neighbors
 3: for all neighbors of $P_i$ : $N_i$ do
 4:     $N_i$ stops its extra behaviors, i.e., control loop.
 5:     $N_i$ selects all pre candidates that satisfy the criteria of having a substitute path, taking into account the pre candidate distance and link stress
 6:     $N_i$ sends a message to the pre candidate selected, $C_{ij}$, letting it know it is a pre candidate.
 7:     for all pre candidates : $C_{ij}$ do
 8:         $C_{ij}$ stops its extra behaviors, i.e., control loop.
 9:         $C_{ij}$ actives the live migration Listener
10:     $N_i$ sends a message to the affected router, $P_i$, informing the pre candidate list
11: $P_i$ receives a pre candidates list from its neighbors
12: $P_i$ selects the best match
13: for all virtual machines hosted in $P_i$ : $V_i$ do
14:     Live migrates $V_i$ to the destination host.
```

To evaluate this scenario, we first set up a virtual environment, as depicted in Figure 7.2 (a), with a user's data flow coming from V_a, passing through V_b and arriving at V_c. In addition, in order to force a link degradation, we set up a traffic generator, which sends a large amount of traffic to the P_b, the physical host of V_b. We expect that after a short period the agent responsible for the affected machine, Pa_b, through its adaptive functionality would trigger the Adaptive Live Migration plan, as it needs to guarantee that the user's request would not experience a request degradation.

Assuming the experiment topology, right after the Pa_b decides to migrate its guest V_b, along with its agents, links and services, Pa_b would have two choices for a destination host, either P_d or P_e. It was such a decision that was evaluated in this first scenario. In this first scenario the nature of the network topology seems to be simple, and since the adaptive live migration plans count on two possible destination hosts, they enable us to test each aspect of our proposed model (Self-organizing, Self-tuning and Knowledge Process). Our aim was to separate each aspect and test this scenario by changing the environment and the local setup conditions to evaluate each feature individually.

A Message Exchange Analysis

Even before an adaptive plan is triggered, all agents from a cluster exchange messages in order to update and share knowledge among such agents. We strongly believe that this process enables the provisioning itself to be a lower cost method of message exchange. When the network appears stable, messages are exchanged frequently to update the knowledge of all devices. Once a critical scenario has been detected, the agents of the cluster in which the affected node belongs halt all other secondary activities, including update requests. Then, they collaborate with each other to select a best match as a destination host. At this moment, since the network might be unstable and vulnerable because of some overloaded links, the message exchange should cost as little in link capacity as possible. Being aware of this special case, we have evaluated the adaptive live migration plan, at first with no differences among agents, in which each agent has the same set up and tuning. As result, to migrate the V_b from P_b to either P_d or P_e, takes less than seven seconds after detection of the critical scenario until the virtual router is fully migrated and the total number of messages exchanged is eight.

Since the live migration module exhibits a certain pattern while executing the adaptive plan, we strongly believe this pattern provides a good template to be applied in simulations of larger networks. In order to measure an initial estimate of the number of message exchanges in our approach for larger environments, we simulated the same behavior of the real experiment in a virtual environment to determine the number of messages that are exchanged, while our approach executes the adaptation itself. The results are depicted in Figure 7.3, in which live migration is triggered from a link coming from R_a to R_b. Assuming a link from R_a to R_b, and with $R_{central}$ as the affected node, we have to consider: (a) a one router radius, (from $R_{central}$ to either R_a or R_b), where eight messages were exchanged; (b) a two router radius, where fourteen messages were exchanged; and (c) a three router radius, where twenty messages were needed. We highlight that, in example (a), for instance, if the topology contained more routers beyond the path R_a to R_b, i.e, more possible paths from R_a to R_b, Figure 7.2 (b), the number of messages would remain the same. This result shows that our proposed solution, even if the network becomes larger, still presents a constant or linear increase depending on the topology.

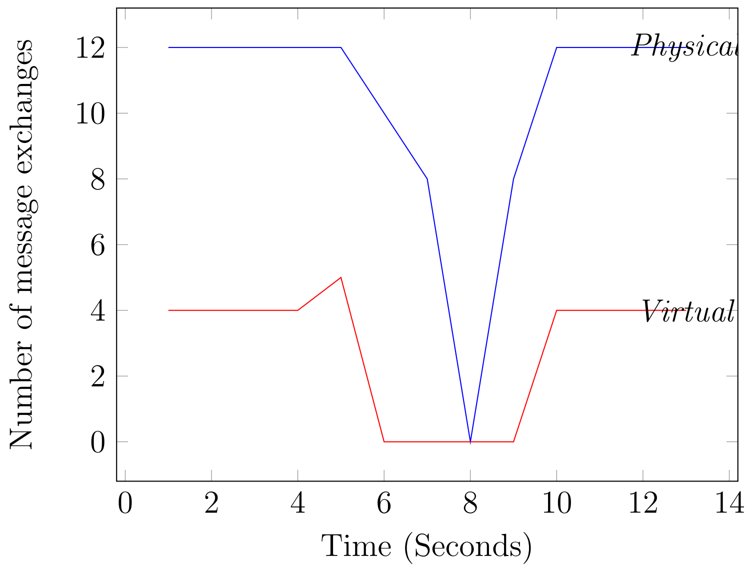


Figure 7.6: Migrate VR One Cluster: Message Exchange Analysis

In Figure 7.6, we plot the number of message exchange needed in the case of physical router overload. Considering a virtual/physical topology as depicted in Figure 7.3 (a), the number of messages needed in a stable scenario is about four for the virtual network, and twelve for the physical one. When the affected agent, $\mathrm{Pa_b}$ anticipates a physical router overload, it stops sending keep-alive messages to its neighbors. As a consequence, its neighbors infer the occurring of adaptation and stop sending messages to the affected node as well. Thus, during the execution of the adaptive plan, the agents from the affected cluster stop sending keep-alive messages and start exchanging supportive messages. Therefore, in this situation, the messages of the physical routers drop from twelve to eight, while the messages of the virtual clusters drop to zero, considering the topology depicted in Figure 7.3.

If we scale up our experiment, consider a topology comprising 3 clusters, in which the physical network contains thirteen physical routers and the virtual topology, also comprising of 3 clusters, with seven virtual routers, as depicted in Figure 7.7. This scenario is the same as the Figure 7.6. The difference, however, relies on the number of needed messages, as depicted in Figure 7.8. In this topology, the number of messages to carry on a stable network is thirty six for the physical network and twelve for the virtual one. When the affected agent detects a physical router overload; the physical network drops the number of messages to thirty two, while the virtual network drops its messages to eight.

Note that, the number of messages in both cases are different while facing an adaptive event. This difference is justified because of the local property. In Figure 7.8 we have three different clusters for both networks. When one cluster detects a critical scenario, it stops sending keep-alive messages to its members.

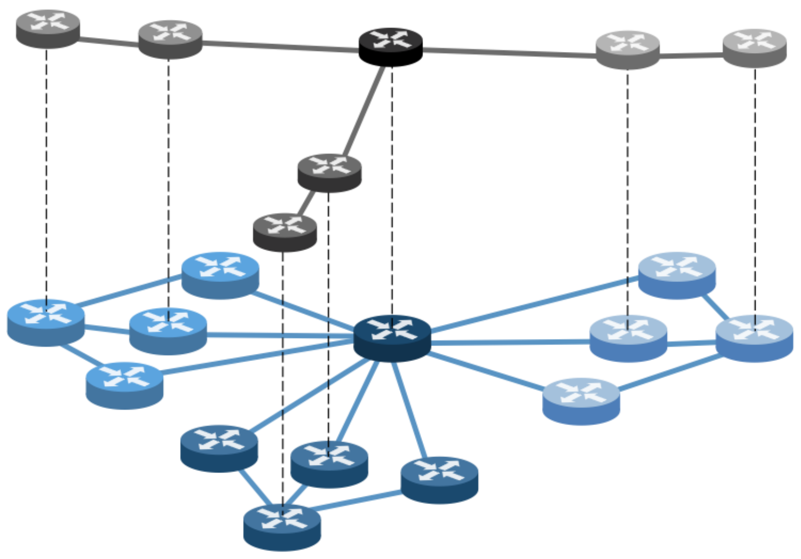


Figure 7.7: Virtual Network with Multiple Clusters

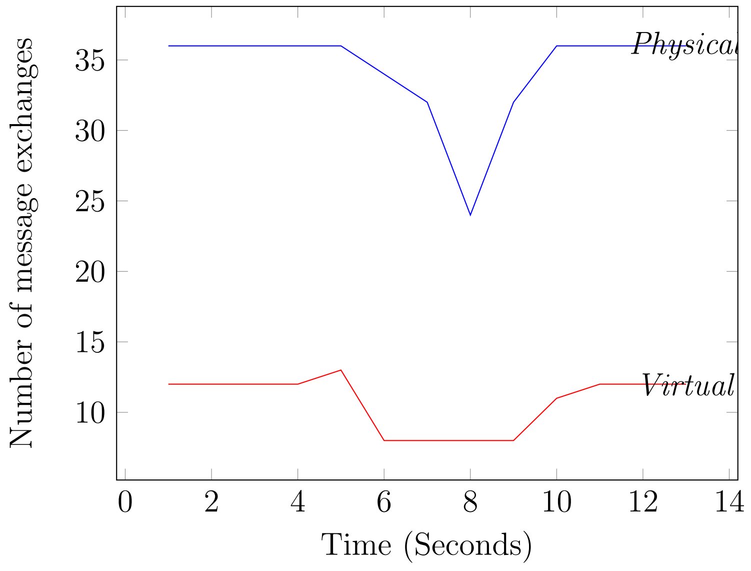


Figure 7.8: Migrate VR 3 Clusters: Message Exchange Analysis

The other two clusters, however, keep running their own management tasks (keep-alive messages).

A knowledge acquiring analysis

The authors of (Samaan e Karmouch, 2009) have raised an issue related to the challenge involved in developing architectures that can host autonomic solutions and coordinate the distributed interactions. In order to decrease the difficulty of maintaining a fully distributed VN, since we believe that the distributed approach has advantages over traditional centralized solutions, we have aggregated the concept of knowledge acquisition /sharing. This approach is also an attempt to decrease the need for exchanging messages to update knowledge at the precise moment the knowledge is required, but rather upon the execution or participation of an adaptive plan.

To understand better the role of knowledge acquisition/sharing in our approach, we have addressed three different experiments with the same adaptive live migration scenario. The main idea behind this set of experiments is to vary the degree of knowledge discovery so as to determine the impact of such a feature at the global level of the proposed model. The following variations have been covered: (i) knowledge inferring through environmental observations with knowledge sharing; (ii) individual knowledge inferring without knowledge sharing; and (iii) no knowledge inferring/sharing with a centralized approach. Like the first evaluation, we have also simulated the same experiment for larger networks. Considering the topology of Figure 7.3 (b) (R_{ij}), the results are depicted in Figure 7.9.

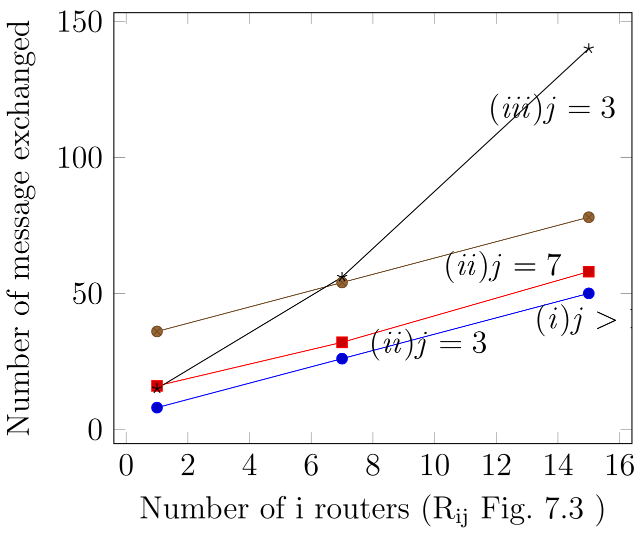


Figure 7.9: A knowledge and scalability Analysis

A Self-tuning/Norms analysis

Self-tuning is a function highly related to how Norms are applied to the model. Self-tuning occurs whenever an agent disobeys a specific norm, and the agents goal is to enable fine-tuning of its own parameters in order to self-adapt based on its local environment. Because of the distributed nature of the environment, there are different clusters, with different environmental conditions and needs and, as a consequence, a different setup would be required. As we are not able to control every single router on the network, we aim, through self-tuning, to tune each agent according to variations in its local environment.

An important function of our model with respect to efficiency is autonomic planning and execution. Since every agent frequently requires the operation of the Monitor and Analyzer functions to support an efficient triggering of adaptive plans better, we have added the Self-tuning concept. Through Self-tuning, we aim to increase the flexibility of the control loop roles in either selecting a better match to support the adaptation task or by analyzing the current network state differently. This concept is an important feature if we take into account that a unique agent has several functions running concurrently, and its main purpose is to be able to express the time and the frequency of each task run in terms of the need for such functionality. Therefore, the question that remains is: if we are dynamically changing the tuning of each agent and its respective tasks, how can we safely guarantee that the tuning will lead to an efficient setup?

If we can clearly recognize a good application of Self-tuning, in which an agent can improve its task allocations, we still need to validate the consequences that would emerge from these experimental scenarios upon applying this concept. In order to address this question, we have evaluated the previous scenario, varying some local aspects of the individual virtual/physical agent, mainly regarding the initial control loop tuning. We started by running two tests: (a) all adaptive agents of the environment have the same initial tuning; and (b) the physical router P_d runs its control loop twice slower than the router P_e, even though all other specifications remain the same, which leads to an initial timeout for the request response. Refer to the Table 7.2 for results.

Case	AP#1	AP#2	AP#3	AP#4
a	P_a	P_b	P_a	P_a
b	time out	time out	P_b	P_b

Table 7.2: Consecutive Adaptive Plans (AP) and its pre candidates selection

Scalability Analysis

In order to address the scalability question, we evaluated the scalability of the distributed live virtual router migration module in our real network and also by simulating that router in a larger network. Since the adaptation plan follows a certain pattern, we used this pattern as a template in a virtual scenario, in which we applied the steps of our candidate discovery and live migration plan in order to measure the number of messages exchanged. This measurement was used to (a) confirm knowledge update, (b) select a good candidate and (c) realize the migration itself. Assuming that the VN's knowledge is always updated, we transferred the experiment to a virtual plan, in order to estimate the metrics for larger networks. In the virtual simulation, we set up a full mesh of substrate topologies with different sizes, from 10 to 60 virtual nodes. Refer to Figure 7.9 for results.

7.2.3
Replace Virtual Router scenario

The replacement of virtual router adaptive plan is triggered in a specific scenario where a virtual router suffers from anomalies and failures such as lack of resource, link overload or whenever it gets unresponsive. To be triggered, the environment condition has to overcome some specified threshold, based on network metrics related to link/CPU/Memory RAM usage.

Once the agent has detected an environment condition that leads to a replacement of virtual routers, it triggers the adaptive plan responsible for replacing the affected virtual router for a new one, capable of handling the current network demand. At this stage, the agent creates a new virtual router, with higher technical specifications, capable of handling the current demand and users requests. Hence, once the new virtual router is created, the agent autonomically swaps the affected virtual router by the created one before the link service exhibit any anomaly to the end users. Furthermore, at this stage, when the virtual router takes place of the affected node, all services, flows and management system are kept running inside the new virtual router.

Thus, in terms of management, to trigger the replacement of the virtual router, the affected agent has to first (i) anticipates a critical scenario, (ii) stops sending keep alive message to its neighbors, (iii) sends a request message to the agent of its physical host, (iv) waits for the update to send an inform message to its neighbors. The adaptation plan is completed when the affected agent goes through all steps, and restart sending a keep alive to its neighbors. From this step on, all keep alive messages are normally exchanged among the virtual routers.

A Self-tuning/Norms analysis

When an agent is anticipating a critical scenario, the set of active norms are the *pre-adaptation plans*. An example of anticipating an event is the advancing of virtual router overload. This pre-adaptation set contains all norms to be fulfilled for the context of anticipating an adaptation event, in which the agent has to detect the problem source and propose all possible solutions. Thus, as for an illustrated example, consider a virtual network, as depicted in Figure 7.3, where the virtual router V_c, anticipates a scenario of router overload.

Norm 5: Resource Usage Boundary Boundary

Norm Goal: Keep CPU and Link usage variation bellow specific thresholds

Addressees: Virtual Agents;
Context: Anticipating adaptation condition
Reward: Increase the stability metric
Punishment: Executes the self-tuning in order to enable a deeper resource usage analysis.

Figure 7.10: Resource Usage Boundary Norm

In order to better understand the current needs of the network flow at this specific point, the agent of the affect virtual router has to analyze the different aspects of the router and its local environment. Indeed, if the network usage exhibits a high variation at some specific point, the agent's autonomic control loop tends to try to understand why the variation has increased and why it has been occurring. To this end, the norm checker executes the self-tune feature in order to enable a richer analyzer, which will, in turn, analyze different aspects of the network, such as different parameters for CPU, Memory Ram, I/O and Link usage.

Depending on the environment condition while anticipating a critical scenario, the normative control loop enable different monitors to collect different data and different translators to analyze the collected data. Such a dynamism at the ACL level is given through the dynamic adjustment of its own parameters. To this end, this normative autonomic control loop programmability is pre-coded as a set of distributed monitoring norms embedded in the agents.

For the virtual router overload scenario, through the right analysis, the Decision-Making can specify the right tech specification for the new virtual router to be created in order to replace the affected one. It might detect, through the set of specific monitors and analyzers, whether it is a CPU, Memory RAM or a Swap size issue that are changing the current network needs.

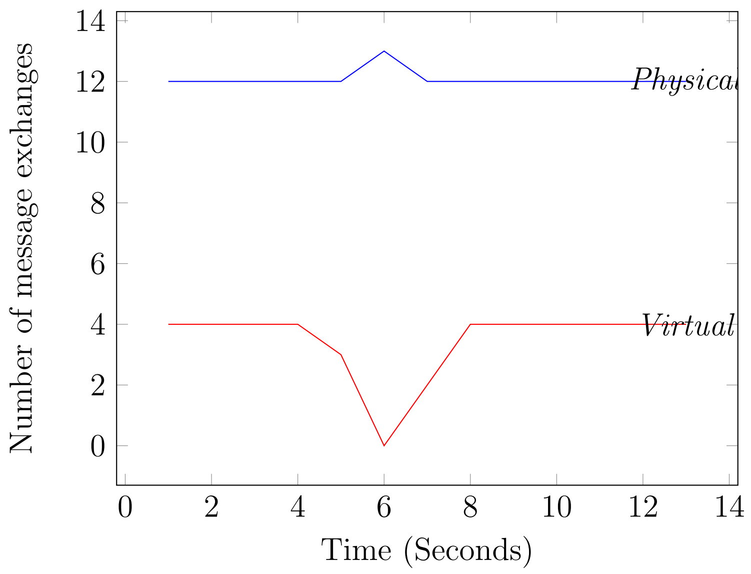


Figure 7.11: Replace VR: A Message Exchange Analysis

A Message exchange analysis

Similarly to Figure 7.6, we plot, in Figure 7.11, the number of message exchange needed in the case of Virtual router overload. Considering a virtual/physical topology as depicted in Figure 7.3 (a), the number of messages needed in a stable scenario is about four for the virtual network, and twelve for the physical one. When the affected agent, Va_c anticipates a virtual router overload, it stops sending keep-alive messages to its neighbors. Thus, during the execution of this adaptive plan, the agents from the affected cluster stop sending keep-alive messages. Unlikely the physical router overload, in the case of virtual overload, there is no need for supportive messages while executing an adaptive plan. Hence, in this scenario, the messages of the physical routers remains twelve, in average, while the messages of the virtual clusters drop to zero, considering the topology depicted in Figure 7.3. The messages of the virtual networks drop to zero, because of the simplicity of the topology used.

7.2.4
Balance Virtual Link scenario

When the virtual network is attending to different users request, i.e., having multiples package paths coming and arriving at different virtual routers, the virtual links usage can present an unbalanced behavior. Such unbalanced performance is characterized by having packages passing through the same links, when there are different alternatives. Having multiple services passing by the same path, might lead to a link overload, and this adaptive plan is responsible for balancing links before it really happens.

In the case a virtual router is having multiple services, it first analyze all the possibilities to balance the network at this point. It first checks (i) the existence of available virtual router which is also capable of handling the user streaming, and (ii) the feasibility of creating a new virtual router, on the same neighborhood, so that it could reassignment one of the streaming path to the new virtual router.

Thus, this adaptive event occurs from the happening of a specific scenario, and might have two different final behaviors: (i) Assign one link routing to another available virtual router; or (ii) create a new virtual router and reassigning the routing load to the new router.

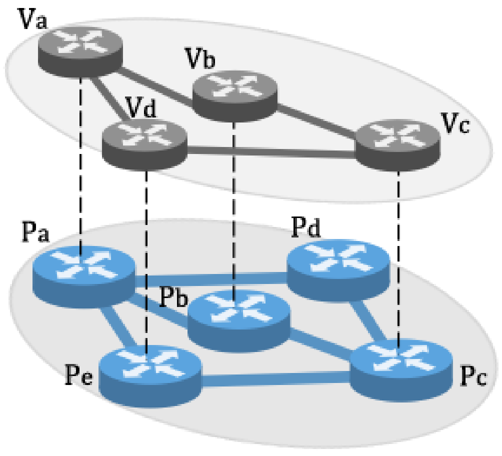


Figure 7.12: Unbalanced Virtual Link Experiment setup

Figure 7.12 shows the experiment setup for the unbalanced virtual links scenario. In this picture, we have multiple links passing through V_c, coming from V_a and arriving at V_b. For this particular experiment, V_d does not carry any package flow. Thus, after Va_c detects this specific situation, it triggers the adaptive avent in order to reassign one of the links from V_c to V_d.

Virtual Links reassigning

At the moment the agents make the decision to balance the virtual network, it sends messages to its neighbors in order to find if there is any candidate to receive one of the service links. To request such an information, the agent sends a request message, checking whether the neighbors are more suitable to be a candidate. To decide the destination in which the link will be reassigned, the agent takes into account, the distance between the starting point and final point of the link service, the availability, the link stress and the request rate of each candidate.

Once the agent decides the final destination, it reassigns the link by changing the link routing. As it is a local behavior with local consequences, only the agents involved in this adaptation plan requires an instant update. The adding of a new virtual router isn't something that impacts the network as a whole, mainly because we are dealing with preconceived virtual networks and services. During the life-time of the network, thought keep-alive messages, the agent involved in this event sends extra information about the new virtual router in the network.

Creating a new Virtual Router

If a virtual router tends to present an overload scenario and the agents detect an unbalanced usage of its links; it first tries to reassign one of the links to another virtual router from the same neighborhood. However, they may not be available for receiving new links connections. In this case, the affected agent will create a new virtual router to be hosted in another physical machine, or on the same physical machine, depending on the network configuration.

Once the new virtual router is up and running, the agent responsible for the affected router, set up a network between the new router and the rest of the nodes, and one of the links is then reassigned to the new router. Just like the Virtual Links reassigning.

A Message exchange analysis

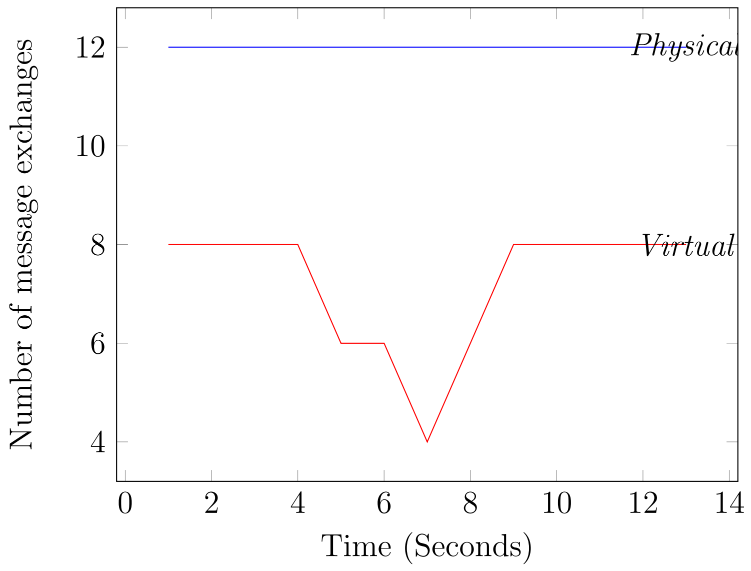


Figure 7.13: Reassign VL: A Message Exchange Analysis

For both scenarios, resigning the affected virtual link by either creating a virtual router or using an existent path, the number of message exchange is, in average, the same as the Replace Virtual router adaptive plan. The difference, however, relies in the fact that, in the situation in which the model has to create a new virtual router and place it on top of the virtual topology, it requires a confirmation message that has to be sent from the affected router to its neighbors. As this confirmation message is not forwarded through the clusters, this adaptive action needs a constant number of message more than the other (where this constant number is equal to the number of neighbors).

We plot, in Figure 7.13, the number of message exchange needed in the case of unbalance virtual links usage. Unlike the previously approximated analysis, this experiment uses a virtual/physical topology, as depicted in Figure 7.12, in which we have a four-virtual routers network. In this topology, the number of messages needed in a stable environment is about eight for the virtual network, and twelve for the physical one. Note, therefore, that the number of virtual routers deployed on top of the substrate node does not affect the number of messages needed on the physical network.

In this case, when the affected agent, Va_c aticipates a virtual link overload (unbalacing link usage), it stops sending keep-alive messages to its neighbors: Va_a and Va_b. Next, during the execution of the adaptive event, the agents from the affected cluster stop sending keep-alive messages to the affected router, Va_c. At this stage, the number of messages on the virtual network drops to four.

Hence, the messages needed for the physical routers maintenance remains twelve, in average, while the messages of the virtual clusters drop to four, considering the topology depicted in Figure 7.12. The messages of the virtual network drops to four because the agents Va_a, Va_b and Va_d keeps communicating with each other. Afterwards, when the adaptation plan is fully executed and, consequently, the affected link is reassigned from $V_a/V_b/V_c$ to $V_a/V_d/V_c$, the number of messages on the virtual network get back to eight.

8 Conclusion and Future Work

In this Master's thesis, we have described the design and validation of an autonomic model for VN provisioning from the MAS perspective. We analyzed the impact and the effectiveness of the self-organizing behavior that emerged from our proposed model, in which it is able to control and manage virtual resources. The experimental results showed that the model satisfies its main goals of (i) automatically reconfiguring itself to meet quality requirements, and (ii) improving the network performance whenever it is exposed to a critical scenario.

8.1 Conclusion

Through our system, we show that it is possible to design an autonomic VN manager by applying an MAS approach together with Self-* capabilities in order to distribute the responsibility to maintain the VN operating in accordance with its policies and requirements. Although our current work focused on the adaptive design, modeling and agent communication, we believe that this general model will certainly support the development of more complex and realistic network structures, which will be able to use adaptive plans according to the environment state. Besides the simple nature of our experimental setup, we evaluated every component applied to our model, showing that they added gains to the infrastructure as a whole, which leads to a linear solution with respect to message exchanges.

We have presented the self-awareness and context-awareness concepts as an important key to enable self-adaptation while enriching Virtual Network management systems. Using context-awareness as a built-in mechanism to facilitate self-adaptation, we ensure that virtual components are able to self-organize themselves. Furthermore, we showed that the ability that autonomic entities have to infer about local and global conditions have increased the effectiveness of the decision making process, in which the agents can better analyze its own condition in terms of network usage and also distinguish in which direction there is a higher overflow. Moreover, we highlight that the triggers of any self-organizing action are the local measurement of the surrounding environment and neighbors information, which brings benefits in terms of reduction of traffic loads and overhead in the case of high variance.

8.2 Future Work

As for short-term future research, besides the evaluation of the proposed model in larger networks we intend to address better the limitations of such an approach, the addition, from an MAS perspective, of the Reputation concept to support the self-tuning and norms checking functionalities. Furthermore, based on a network analysis, we highlight the need to specify proper boundaries to control the decision about the network state. In addition, we recently performed an evaluation of other adaptive plans besides the adaptive live migration router, which will be used to enrich the analysis of all aggregated paradigms of our model. As for adaptive plans, we highlight the balancing of virtual links by deploying new virtual routers to the virtual topology as well as balancing links by using existing virtual routers and replacement of virtual routers.

Finally, for the Normative autonomic control loop proposed in this thesis, we aim to generalize the normative autonomic control loop to cope with arbitrary plans and behaviors, as this would allow us to address more complex and self-adaptive applications. We would also like to extend the concept of the normative autonomic control loop in the sense of having adaptive norms, besides having a normative system to support self-adaptation behavior, with dynamic thresholds and constraints dependent on the environmental context. In this case, each member of the set of available norms is able to self-adapt and evolve during the life-time of the system. We also want to formalize the concept of normative autonomic control loop and investigate the verification of norms in a self-adaptive system by detecting obligations that cannot be fulfilled, and prohibitions that will prevent adaptive events such as inconsistencies and conflicts.

9
Bibliography

ANDERSON, T. et al. Overcoming the internet impasse through virtualization. **Computer**, v. 38, n. 4, p. 34–41, 2005. ISSN 0018-9162.

ARCHITECTURE and Design for the Future Internet, 4WARD FP7 project. Disponível em: <http://www.4ward- project.eu/>.

AXELROD, R. M. **The complexity of cooperation: Agent-based models of competition and collaboration**. Princeton, NJ: Princeton University Press, 1997. xiv, 232 p p.

BLUMENTHAL, M. S.; CLARK, D. D. Rethinking the design of the internet: The end-to-end arguments vs. the brave new world. **ACM Trans. Internet Technol.**, ACM, New York, NY, USA, v. 1, n. 1, p. 70–109, ago. 2001. ISSN 1533-5399. Disponível em: <http://doi.acm.org/10.1145/383034.383037>.

BOELLA, G.; TORRE, L. van der. From the theory of mind to the construction of social reality. In: **Procs. of Annual Conference on the Cognitive Science Society**. Mahwah (NJ): Lawrence Erlbaum, 2005. p. 298–303. Disponível em: <http://icr.uni.lu/leonvandertorre/papers/cogsci05.pdf>.

BOELLA, G.; TORRE, L. W. N. van der. Regulative and constitutive norms in normative multiagent systems. In: **KR**. [s.n.], 2004. p. 255–266. Disponível em: <http://icr.uni.lu/leonvandertorre/papers/kr04.pdf>.

BOUTABA, R.; POLYRAKIS, A. Projecting advanced enterprise network and service management to active networks. **IEEE Netw**, v. 16, n. 1, p. 28–33, 02 2002.

CALO, S.; SLOMAN, M. Guest editorial: Policy-based management of networks and services. **Journal of Network and Systems Management**, Kluwer Academic Publishers-Plenum Publishers, v. 11, n. 3, p. 249–252, 2003. ISSN 1064-7570.

CAMAZINE, S. et al. **Self-Organization in Biological Systems**. Princeton, NJ, USA: Princeton University Press, 2001. ISBN 0691012113.

CHENG, X. et al. Virtual network embedding through topology-aware node ranking. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 41, n. 2, p. 38–47, abr. 2011. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1971162.1971168>.

CLARK, D. et al. **New Arch: Future Generation Internet Architecture**. [S.l.], 2004. Disponível em: <http://www.isi.edu/newarch/iDOCS/final.finalreport.pdf>.

COMPUTING, A. et al. An architectural blueprint for autonomic computing. **IBM White Paper**, 2006.

CONTE, R.; CASTELFRANCHI, C. **Cognitive and Social Action**. [S.l.]: Routledge, 1995. ISBN 1857281861.

DIGNUM, F. Autonomous agents with norms. **Artificial Intelligence and Law**, Kluwer Academic Publishers, v. 7, n. 1, p. 69–79, 1999. ISSN 0924-8463.

EGI, N. et al. Evaluating xen for router virtualization. In: **ICCCN**. [S.l.]: IEEE, 2007. p. 1256–1261. ISBN 978-1-4244-1251-8.

EGI, N. et al. Evaluating xen for router virtualization. In: **Computer Communications and Networks. ICCCN 2007. Proceedings of 16th International Conference on**. [S.l.: s.n.], 2007. p. 1256–1261. ISSN 1095-2055.

FERNANDES, N. C. et al. Virtual networks: isolation, performance, and trends. **Annales des Télécommunications**, v. 66, n. 5-6, p. 339–355, 2011.

GARCíA-CAMINO, A. et al. Constraint rule-based programming of norms for electronic institutions. **Autonomous Agents and Multi-Agent Systems**, Springer US, v. 18, n. 1, p. 186–217, 2009. ISSN 1387-2532. Disponível em: <http://dx.doi.org/10.1007/s10458-008-9059-4>.

HORN, P. **Autonomic Computing: IBM's Perspective on the State of Information Technology**. [S.l.], 2001.

HOUIDI, I. et al. Virtual network provisioning across multiple substrate networks. **Computer Networks**, v. 55, n. 4, p. 1011 – 1023, 2011. ISSN 1389-1286. Special Issue on Architectures and Protocols for the Future Internet. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128610003786>.

HOUIDI, I.; LOUATI, W.; ZEGHLACHE, D. A distributed and autonomic virtual network mapping framework. In: **Autonomic and Autonomous Systems, ICAS 2008. Fourth International Conference on**. [S.l.: s.n.], 2008. p. 241–247.

HOUIDI, I.; LOUATI, W.; ZEGHLACHE, D. A distributed and autonomic virtual network mapping framework. In: **Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems**. Washington, DC, USA: IEEE Computer Society, 2008. (ICAS '08), p. 241–247. ISBN 978-0-7695-3093-2. Disponível em: <http://dx.doi.org/10.1109/ICAS.2008.40>.

HOUIDI, I. et al. Adaptive virtual network provisioning. In: **Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures**. ACM, 2010. (VISA '10), p. 41–48. ISBN 978-1-4503-0199-2. Disponível em: <http://doi.acm.org/10.1145/1851399.1851407>.

IBM (Ed.). **An Architectural Blueprint for Autonomic Computing**. [S.l.], jun. 2005.

KAMAMURA, S. et al. Control and visualization system for managed self-organization network. In: **Network and Service Management (CNSM), 2011 7th International Conference on**. [S.l.: s.n.], 2011. p. 1–4.

KANELLOPOULOS, D. N. Ontology-driven knowledge management for cognitive networks. **International Journal of Enterprise Network Management**, v. 4, n. 3, p. 229–246, Jan 2011.

LóPEZ, F. L. y; LUCK, M. Modelling norms for autonomous agents. In: **ENC**. [S.l.]: IEEE Computer Society, 2003. p. 238–245. ISBN 0-7695-1915-6.

LóPEZ, F. L. y; LUCK, M.; D'INVERNO, M. Constraining autonomy through norms. In: **Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2**. New York, NY, USA: ACM, 2002. (AAMAS '02), p. 674–681. ISBN 1-58113-480-0. Disponível em: <http://doi.acm.org/10.1145/544862.544905>.

LOPEZ, F. Lopez y; LUCK, M.; D'INVERNO, M. Normative agent reasoning in dynamic societies. In: **Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2**. Washington, DC, USA: IEEE Computer Society, 2004. (AAMAS '04), p. 732–739. ISBN 1-58113-864-4. Disponível em: <http://dx.doi.org/10.1109/AAMAS.2004.197>.

MAGEDANZ, T.; KARMOUCH, A. Mobile software agents for telecommunication applications. **Computer Communications**, v. 23, n. 8, p. 705–707, 2000.

MARQUEZAN, C. et al. Distributed autonomic resource management for network virtualization. In: **Network Operations and Management Symposium (NOMS), 2010 IEEE**. [S.l.: s.n.], 2010. p. 463–470. ISSN 1542-1201.

MIYAMURA, T. et al. Dynamic resource allocation mechanism for managed self-organization. In: **Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific**. [S.l.: s.n.], 2011. p. 1–4.

MOVAHEDI, Z. et al. A survey of autonomic network architectures and evaluation criteria. **Communications Surveys Tutorials, IEEE**, v. 14, n. 2, p. 464–490, 2012. ISSN 1553-877X.

O'HARE, G. M. P.; JENNINGS, N. R. **Foundations of distributed artificial intelligence.** [S.l.]: Wiley, 1996. I-XIV, 1-576 p. (Sixth-generation computer technology series). ISBN 978-0-471-00675-6.

OLSSON, R. pktgen the linux packet generator. In: **Proceedings of the 2005 Ottawa Linux Symposium**. [S.l.: s.n.], 2005. v. 2.

PISA, P. et al. Openflow and xen-based virtual network migration. In: PONT, A.; PUJOLLE, G.; RAGHAVAN, S. (Ed.). **Communications: Wireless in Developing Countries and Networks of the Future**. [S.l.]: Springer Berlin Heidelberg, 2010, (IFIP Advances in Information and Communication Technology, v. 327). p. 170–181. ISBN 978-3-642-15475-1.

PREHOFER, C.; BETTSTETTER, C. Self-organization in communication networks: principles and design paradigms. **Communications Magazine, IEEE**, v. 43, n. 7, p. 78–85, 2005. ISSN 0163-6804.

RUTH, P. et al. Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. In: **Autonomic Computing, 2006. ICAC '06. IEEE International Conference on**. [S.l.: s.n.], 2006. p. 5–14.

SAMAAN, N.; KARMOUCH, A. Towards autonomic network management: an analysis of current and future research directions. **Communications Surveys Tutorials, IEEE**, v. 11, n. 3, p. 22–36, 2009. ISSN 1553-877X.

SENNA DANIEL M. BATISTA, M. A. S. J. C. R.; MADEIRA, E. R. M. Experiments with a self-management system for virtual networks. In: **Anais / II Workshop de Pesquisa Experimental da Internet do Futuro**. Campo Grande, MS, Brazil: [s.n.], 2011. p. 7–10. ISBN 2177-496X. Disponível em: <http://sbrc2011.facom.ufms.br/files/workshops/wpeif/wpeif.pdf>.

SERUGENDO, G. D. M.; GLEIZES, M. P.; KARAGEORGOS, A. Self-organisation and emergence in mas: An overview. **Informatica (Slovenia)**, v. 30, n. 1, p. 45–54, 2006.

SERUGENDO, G. M. et al. Self-organisation: Paradigms and applications. In: SERUGENDO, G. M. et al. (Ed.). **Engineering Self-Organising Systems**. [S.l.]: Springer Berlin Heidelberg, 2004, (Lecture Notes in Computer Science, v. 2977). p. 1–19. ISBN 978-3-540-21201-0.

SPIVEY, J. M. **The Z Notation: A Reference Manual**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989. ISBN 0-13-983768-X.

TESAURO, G. et al. A multi-agent systems approach to autonomic computing. In: **Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.** [S.l.: s.n.], 2004. p. 464–471.

TINNEMEIER, N.; DASTANI, M.; MEYER, J.-J. Roles and norms for programming agent organizations. In: **Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1**. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009. (AAMAS '09), p. 121–128. ISBN 978-0-9817381-6-1. Disponível em: <http://dl.acm.org/citation.cfm?id=1558013.1558029>.

TUOMELA, R. **The Importance of Us: A Philosophical Study of Basic Social Notions**. [S.l.]: Stanford University Press, 1995.

WINTER, R.; SCHILLER, J. Crosstalk: A data dissemination-based crosslayer architecture for mobile ad-hoc networks. In: **in Proceedings of ASWN**. [S.l.: s.n.], 2005.

WINTER, R. et al. Crosstalk: cross-layer decision support based on global knowledge. **Communications Magazine, IEEE**, v. 44, n. 1, p. 93–99, Jan 2006. ISSN 0163-6804.

YU, M. et al. Rethinking virtual network embedding: Substrate support for path splitting and migration. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 38, n. 2, p. 17–29, mar. 2008. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1355734.1355737>.

ZAMBONELLI, F. et al. On self-adaptation, self-expression, and self-awareness in autonomic service component ensembles. In: **Proceedings of the 2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops**. Washington, DC, USA: IEEE Computer Society, 2011. (SASOW '11), p. 108–113. ISBN 978-0-7695-4545-5. Disponível em: <http://dx.doi.org/10.1109/SASOW.2011.24>.

ZHU, Y.; AMMAR, M. H. Algorithms for assigning substrate network resources to virtual network components. In: **INFOCOM**. [S.l.]: IEEE, 2006.

A
Details of XEN: The virtualizing tool

Xen is open source industry software for machine virtualization. It offers an efficient, strong and secure feature set for virtualization. The OS supported by Xen are Windows and Linux, which makes it neutral. Being independent, Xen allows Domain 0 to be the unique VM and it has control over all the other VMs.

Xen is a virtualizing tool that provides as built-in features, (i) creation of virtual machines and (ii) live virtual machine migration. The creation of virtual machine is straightforward. To perform live VM migration, however, certain requirements must be fulfilled.

A.1
Creating Virtual Machines

The autonomic creation and instantiation of a new virtual machine on the top of a physical network is an important feature of this work. Thus, whenever an agent makes a decision regarding adaptive plans, the creation of a new virtual machine may be involved. For instance, the adaptive plan responsible for replacing an affected virtual router triggers the creation of a new virtual router to replace the affected one. To this end, we use the instantiation of a virtual machine mechanism offered by XEN capabilities.

XEN standard offers several different ways to create a new guest domain on the top of a physical machine. Even not being the purpose of this master's thesis cover Xen functions, for the sake of optimization, however, we have had to evaluate different ways of doing so, in order to achieve a significant virtual speed-up at instantiation phase of the virtual machine creation task.

Creating a new virtual machine from a previous installation takes two minutes to complete. On the other hand, if we simple copy a virtual machine from a cache repository, it takes less than two seconds to complete. Note that, to validate the creation of a virtual machine, we use Ubuntu 12.4 (without GUI) as the OS. To start a virtual machine, therefore, XEN takes less than one second to starts up a virtual Ubuntu OS.

A.2 Performing Live VM Migration

To perform live VM migration, there are certain requirements to follow. Templates of the virtual machine have to be stored on both hosts. The live VM migration is done easily, when we have OS images stored on the physical machines. Thus, XEN is responsible for copying the current state and all running services of the affected virtual machine to the destination host. On the physical network we provide (Appendix B), the total time of executing a Live VM migration is about 6 seconds.

B
Experimental Environment Details

In order to validade our proposed model, we built a real physical network composed of five machines, specified as follow:

–16GB RAM

–Intel(R) Core(TM) i7- 2600 CPU @ 3.40GHz

–2 terabyte of Hard Drive

–Intel Gigabit ET Dual Port Server Adapter PCI Express

–Ubuntu 12.4

–Xen 4.2 as the virtualization tool. Refer to section XX, for further details.

As already stated, the autonomic monitoring function described on this thesis collects and analyzes different data regarding link and resource usage. They are:

–Link health:
Capacity, Propagation delay, error rate, transmission delay, Packet loss, Packet out of order.

–Operational System resource heath:
CPU, Memory RAM, Fowarding delay, Fowarding capacity, processing delay.

Finally, to evaluate different adaptive cases, we needed to force some network degradation events. To this end, we used a linux testing tool named *Pktgen*[1] (Olsson, 2005), which is included in the Linux kernel. *Pktgen* is used to generate ordinary packets to test network experiments. It specially involves testing of routers or bridges which often also use the Linux network stack. Because *Pktgen* is a “in-kernel” tool it can generate high bandwith and very high packet rates to load routers, bridges, and other network devices.

[1]http://people.kth.se/ danieltt/pktgen/