Technical Report CS-2014-03

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

Carolina Valadares[a], Donald Cowan[b], Carlos Lucena[a], Davy Baía[a]

[a]*Department of Informatics, PUC-Rio, Rio de Janeiro - RJ, Brazil*
[b]*David R. Cheriton School of Computer Science, University of Waterloo, Waterloo - ON, Canada*

Abstract

Recent research in Network Virtualization has focused on the Internet ossification problem [1] whereby multiple independent virtual networks (VN) [1] 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 re-thinking the design and architectural principles of VN [2] [3]. However, to the best of our knowledge, there has been little investigation into the autonomic behavior of such architectures [4][5]. 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

1. Introduction

Network Virtualization (NV) has recently received substantial attention from the academic community. Because of NVs 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, NVs proponents have presented NV as a promising approach to reducing the complexity of managing networks and virtual resources [2][3].

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[6], and responds to users' requests just as any traditional network architecture. In addition VN supports autonomic self-management through adaptation plans which allow the efficient distribution of physical resources among VNs virtual devices.

The virtual network provisioning or adaptive maintenance system aims to deal 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.

Many studies in the Network field [6] have applied Multiagent systems (MAS) and Self-* approaches to support solutions to Network Virtualization management. This paper extends our previous work [7] and discusses management issues that arise in a VN infrastructure from a MAS perspective.

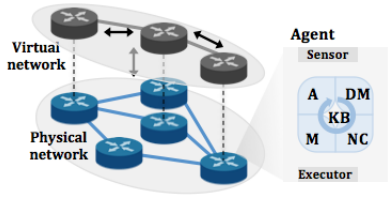


Figure 1: VN Architecture and the agent control loop

We intend to deal with the additional complexity of this new VN concept by enabling autonomic and self-management behavior through Multiagent system and Self-* paradigms. Thus, the main goal of this paper is to fill in the gap that virtualization studies exhibit by covering design principles, from the autonomic computing perspective [8], in terms of autonomic distribution, means of communication, degree of intelligence and independence. Furthermore, 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.

Thus, 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.

To this end, we designed and evaluated a distributed, autonomic and self-organizing system strongly based on MAS and Self-Organizing approaches to ensure distributed negotiation and synchronization between the substrate (physical) nodes and virtual resources. 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. We apply such concepts (MAS and Self-*) to enable communication between the physical and virtual agents to provide performance and scalability of the distributed and autonomic VN manager over traditional approaches.

As our proof-of-concept, we implemented and validated, through testbed experiments, a Virtual Network Provisioning System (VNPS) wherein the VNPS can act upon a critical scenario and re-organize the VN. We showed that, in the case of physical router degradation, our model is able to anticipate a possible future failure. Such a failure because of the lack of physical resources, triggers the adaptive live migration of virtual routers and executes the adaptive plan in a matter of seconds. This adaptation involves the exchange of eight messages in a Network, as shown in Figure 1. 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 almost linearly. We then 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.

2. Related Work

To the best of our knowledge, there are few studies on adaptive provisioning of instantiated VNs to cope with dynamic changes in service demands and resource availability from the MAS perspective [9] [10]. Many approaches to solving the VN provisioning problem in the Network community have dealt with virtual node live migration to a distinct host. There are also a few projects dealing with VN management in general from a MAS perspective.

The authors of [11] 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. Similarly, our previous work [7], proposed an initial adaptive virtual resource provisioning, which brings substrate node agents to cope with failures and severe performance degradation in network virtualization. Such proposals as just described are highly detailed in terms of Network architecture and functionality; but, we noticed there is a lack of information regarding autonomic behavior, MAS and Self-* capabilities.

[9] proposes a distributed self-organizing model based on MAS properties to manage the substrate network resources in the case of failures and link degradation. This paper highlights Self-* properties and briefly describes how the autonomic entities communicate and cooperate to maintain the virtual network's stability and high levels of functionality. Furthermore, there is also the Autonomic Networks field, which deals with Self-* paradigms applied

on autonomic networks, such as described in [4] and [12]. In [4], the authors propose four self-organizing paradigms to be applied in autonomic networks, while the authors of [12] describe a survey covering all possible self-organizing and self-aware properties in VNs; they also give an overview of knowledge retrieval and sharing.

We also note that these approaches have treated the VN provisioning challenge from a semi-decentralized way, in which the autonomic entities are only spread over substrate nodes. In fact some of the autonomic entities are even centralized[11]. In contrast, the self-organizing model proposed in this paper addresses the management of network resources by taking advantage of the total distribution of the autonomic entities throughout the network, including virtual networks, and their components rather than only physical routers.

3. Approaching Self-Adaptation

The VNPS 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 [13]. This solution is capable of creating multiple customized VNs and at the same time it exhibits flexible management and real-time control.

Our adaptive model is based on a distributed algorithm, which embeds an autonomic agent 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, as a consequence, 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 VN emerging as a self-organizing entity. The agent is in charge of handling local behavior to enable proper control and management of the virtual network, its components and the network flows. Thus, this control and management maintains the efficient use of physical resources for network virtualization. Assuming that the VN has already been provided, our adaptive model includes the following features: (i) self-adaptive behavior; (ii) resource and context-awareness; and (iii) knowledge acquisition/sharing. As a consequence of these features acting in combination, 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.

3.1. 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 models or paradigms.

In this section, we address the following key questions related to a networks self-adaptive properties: (i) What principles do self-adaptive systems and autonomic VN provisioning have in common? and (ii) 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 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.

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 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. These rules are: (i) maintain the usage of the link at most at a determined threshold; (ii) maintain a balanced link usage; and (iii) keep local knowledge current.

The second property that we highlight, which relates our problem to self-organization [4], 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 our model, which uses implicit coordination, self-organization, context and global knowledge are inferred from the 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 a 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 property applied to our autonomic model is related to obtaining and keeping local network state information. The author of [4] 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 some 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.

3.1.1. Control Loop

The ability of the autonomic agents to react to changes in the network is provided by the adaptation of the Control Loop concept, defined by IBM for autonomic computing [14]. 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 there exists distinct transitions 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.

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 networks 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.

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 than a link with medium usage but with increasingly more requests.

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. Norm Checking

The last function of the control loop besides 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 Self-Tuning for further details on Norms and how they are checked.

3.2. Resource and context awareness

The author of [12] believes that building a knowledge process to acquire/share knowledge as a built-in reasoning mechanism is a key concept to achieving autonomic management to describe better the precise model of the management system. Following such a belief and aiming to enable cooperation with some self-organizing characteristics already highlighted in this paper, we have added the concept of context-awareness, knowledge discovery and a knowledge-sharing process in an attempt to increase our system's degree of intelligence. Like [12], we also believe that the ability of the autonomic entities 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.

Thus, we enabled self-awareness through: (i) knowledge representation, which can be characterized as behavioral, structural and adaptive, all represented by ontologies; and (ii) knowledge acquisition/sharing, in accordance with self-organizing paradigms. In the latter each entity is able to infer local conditions through observations and message exchanges in its neighborhood.

1. Knowledge Representation

Behavioral Knowledge refers to domain knowledge, in which it represents the different behaviors of the system, its properties, its environmental conditions, and its relationships as well as the representation of the description of each behavior pattern that our model exhibits. An example of behavioral knowledge applied in our system is the relation between norms and the consequence of accepting or declining a specific norm.

Structural Knowledge like behavioral knowledge, is also related to domain knowledge. The difference is that structural knowledge represents a description of the physical network itself, the virtual network and its components. An example of such knowledge is, for instance, the physical machine resources available and the network topology, as well as the possible resource capability of each virtual device.

Adaptive Knowledge describes the adaptive plans, the conditions in which each adaptive plan occurs and the consequences of such adaptation. It also describes how an agent must behave when it faces an execution of an adaptive plan in the neighborhood. We can represent all rules, conditions and solutions related to each adaptive plan through adaptive knowledge.

2. Knowledge Acquisition

A goal of this model is to reduce dependence on pre-embedded knowledge, as it disturbs the robustness of our model because of the constant need for synchronization among distributed agents. By avoiding duplicates, and the need of frequent synchronization, the proposed model eliminates any explicit knowledge base. At the moment both the physical and the virtual networks are provided, each virtual and physical router is built with the description of its own network setup, and its direct neighbors. Every router must be aware of its neighbors to be able to configure data flows.

After the VN has been defined, the discovery knowledge method runs inside each entity so that each agent is able to listen to the environment and come to a conclusion about the local state. For instance, after the knowledge discovery method has run, the agent knows the virtual topology and the local traffic flows. If frequent keep-alive messages are absent an agent might predict the status of this particular neighbor, which could be participating in an adaptive plan. This knowledge discovery mechanism is an attempt to decrease the message exchange volume, which saves resources in the event of critical scenarios.

Another way of acquiring knowledge is by passing messages through the neighborhood, thus enabling the inference of the global state or a topology change. The process of sharing knowledge consists basically of sharing data recursively among neighbors, in order to let the different neighbor levels know about all peculiar aspects of the network over all the physical/virtual routers.

3.3. Enabling Self-tuning through Norms

Self-tuning is a feature of norms and how they are used by the system. It refers to our model's ability to modify a set of its own parameters dynamically at run time over the VN's lifetime, in order to enable minor adaptation operations. For instance, it can dynamically adapt the control loop's run rate, or even monitor the environment for longer than the normal analysis functionality. Moreover, it is in charge of executing small changes in local parameters to discover a better setup for the autonomic agent.

Furthermore, the norms concept provides each virtual device with a controlled autonomy. By restricting a virtual devices behavior to prevent malfunctions and undesirable behavior, the model also gains a certain degree of dynamism and programmability, since it self-tunes its local parameters according to current network loads at that specific location. Thus, norms and self-tuning are responsible for leading to a more dynamic behavior, in which the core of the self-organizing control loop is able to self-adapt, by fine tuning some of its local parameters in response to environment variations.

As stated before, the autonomic control loop is composed of a norm checker. This component is responsible for checking all pre-determined norms, and applying their respective actions whenever they are accepted. In this sense, the control loop itself is also in charge of controlling the acceptance/refusal of a specific norm. For each possibility, there are the following consequences: (i) for norm acceptance, the norm checker carries out the norm action upon the agent and its environment or (ii) if a norm is refused, the agent's reputation is decreased and self-tuning comes into play. We highlight that "reputation" is a way to classify network agents; hence, one of an agent's goals is to maintain a high reputation.

1. Self-tuning by collaboration norm
Norm #1: Always collaborate with an adaptation plan, by responding to all support requests. Reward N#1: Increase the agents cooperation reputation. Punishment N#1: Decrease the agents cooperation reputation and execute self-tuning in order to decrease the time the management functions (control loop components) take to execute the control loops actions, so that more time is available to respond to requests.

The application of this norm is ideal in cases in which the running time of the control loop functions is higher when compared to the loops idle time, which leads to a lesser availability to respond to other requests. We believe that one reason for an agent not to respond to a request may be because the agent has been busy, because of its highly distributed nature.

4. Experimental Evaluation

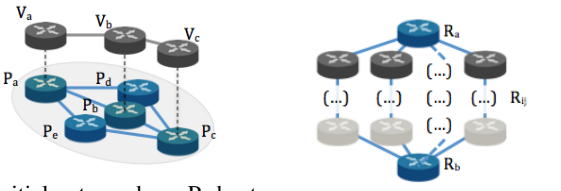
In this section, we provide the environment details as well as the initial experimental results to evaluate the efficiency of the proposed model and its adaptive plans.

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 [9] 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 attempt to test it in a real environment, using a real small physical network composed of six machines. After observing the behavior of such a model on a real network, we validate it by running simulations for larger networks. We also note that the paper focuses on the appropriate choice of an autonomic plan to manage virtual routers and physical resources efficiently by allowing them to re-organize constantly, rather than only in a migration mode.

To this end, we initially set up a VN infrastructure, as depicted in Figure 2, on top of a physical topology. To take advantage of customized virtual machines with different setups, we have used XEN [13] as a virtualized operating system provider to perform the roles of multiple independent virtual machines on a shared substrate.

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.

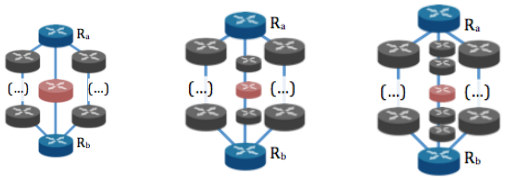
First, we assessed the efficiency of the model, measuring the delay of communication together with the total amount of time incurred to adapt the VN when a virtual or physical device performs poorly or is overloaded. Next,



(a) Initial setup where P_i hosts the agent Pa_i as V_i does Va_i

(b) Possible topologies. (R_{ij})

Figure 2



(a) (b) (c)

Figure 3: Message exchanging analysis

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 composed of the same environment setup as the initial experiment, applying the 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.

In order to cover all features in our proposed solution, we evaluated the main critical scenario that could occur in a VN environment: a physical router overload.

4.1. 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.

To evaluate this scenario, we first set up a virtual environment, as depicted in Figure 2, 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.

4.1.1. 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 six 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 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, (Fig 2b) the number of messages would remain the same, eight. This result shows that our proposed solution, even if the network becomes larger, still presents a constant or linear increase depending on the topology.

4.1.2. A knowledge acquiring analysis

The authors of [12] 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 2b (R_{ij}), the results are depicted in Figure 4.

4.1.3. 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

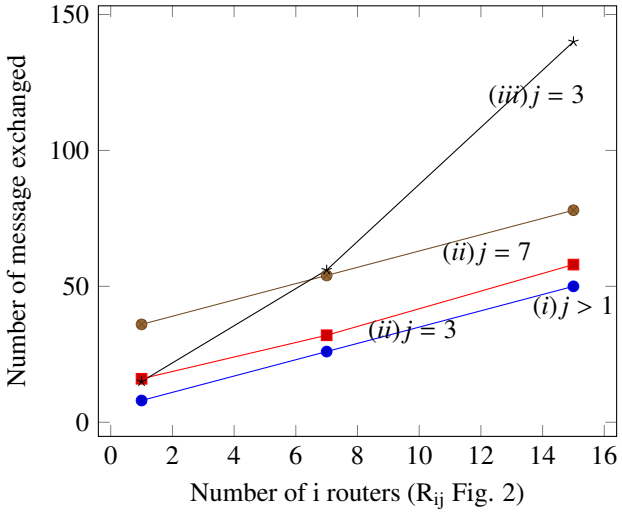


Figure 4: A knowledge and scalability Analysis

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 1 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 1: Consecutive Adaptive Plans (AP) and its pre candidates selection

4.1.4. 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 4 for results.

5. Conclusions

In this paper we have described the design and validation of an autonomic model 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 it satisfies the model's main goal of automatically reconfiguring itself to meet quality requirements and to improve the network performance whenever exposed to a critical scenario.

Through our system, we show that it is possible to design an autonomic VN manager by applying a 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.

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 a 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.

References

[1] T. Anderson, L. Peterson, S. Shenker, J. Turner, Overcoming the internet impasse through virtualization, Computer 38 (4) (2005) 34–41. doi:10.1109/MC.2005.136.

[2] M. S. Blumenthal, D. D. Clark, Rethinking the design of the internet: the end-to-end arguments vs. the brave new world, ACM Trans. Internet Technol. 1 (1) (2001) 70–109. doi:10.1145/383034.383037.
URL http://doi.acm.org/10.1145/383034.383037

[3] I. Houidi, W. Louati, D. Zeghlache, A distributed and autonomic virtual network mapping framework, in: Autonomic and Autonomous Systems, ICAS 2008. Fourth International Conference on, 2008, pp. 241–247. doi:10.1109/ICAS.2008.40.

[4] C. Prehofer, C. Bettstetter, Self-organization in communication networks: principles and design paradigms, Communications Magazine, IEEE 43 (7) (2005) 78–85. doi:10.1109/MCOM.2005.1470824.

[5] Z. Movahedi, M. Ayari, R. Langar, G. Pujolle, A survey of autonomic network architectures and evaluation criteria, Communications Surveys Tutorials, IEEE 14 (2) (2012) 464–490. doi:10.1109/SURV.2011.042711.00078.

[6] C. Marquezan, L. Granville, G. Nunzi, M. Brunner, Distributed autonomic resource management for network virtualization, in: Network Operations and Management Symposium (NOMS), 2010 IEEE, 2010, pp. 463–470. doi:10.1109/NOMS.2010.5488490.

[7] C. Valadares, M. Netto, C. Lucena, A normative and self-organizing piloting model for virtual network management, in: Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações, IEEE, 2013, pp. 41–46.

[8] G. Tesauro, D. Chess, W. Walsh, R. Das, A. Segal, I. Whalley, J. Kephart, S. White, A multi-agent systems approach to autonomic computing, in: Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004., 2004, pp. 464–471.

[9] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, L. Mathy, Adaptive virtual network provisioning, in: Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, VISA '10, ACM, 2010, pp. 41–48. doi:10.1145/1851399.1851407.
URL http://doi.acm.org/10.1145/1851399.1851407

[10] I. Fajjari, M. Ayari, O. Braham, G. Pujolle, H. Zimmermann, Towards an autonomic piloting virtual network architecture, in: New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on, 2011, pp. 1–5. doi:10.1109/NTMS.2011.5720639.

[11] P. Ruth, J. Rhee, D. Xu, R. Kennell, S. Goasguen, Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure, in: Autonomic Computing, 2006. ICAC '06. IEEE International Conference on, 2006, pp. 5–14. doi:10.1109/ICAC.2006.1662376.

[12] N. Samaan, A. Karmouch, Towards autonomic network management: an analysis of current and future research directions, Communications Surveys Tutorials, IEEE 11 (3) (2009) 22–36. doi:10.1109/SURV.2009.090303.

[13] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, T. Schooley, Evaluating xen for router virtualization, in: Computer Communications and Networks. ICCCN 2007. Proceedings of 16th International Conference on, 2007, pp. 1256–1261. doi:10.1109/ICCCN.2007.4317993.

[14] A. Computing, et al., An architectural blueprint for autonomic computing, IBM White Paper.