AAVP: An Innovative Autonomic Architecture for Virtual network Piloting

Ilhem FAJJARI
LIP6 Laboratory, University of Paris VI, Paris, France
Othmen BRAHAM
LIP6 Laboratory, University of Paris VI, Paris, France
Mouna AYARI
CRISTAL lab, National School of Computer Sciences, University of Manouba, Tunisia
Guy PUJOLLE
LIP6 Laboratory, University of Paris VI, Paris, France
and Hubert ZIMMERMANN
Ginkgo Networks Company, Montrouge, France

In the few last years, the Internet has known an impressive success. This success has stimulated the development and the deployment of new technologies and advances applications. Nonetheless, due to its size and scope, this large network has become victim of its own success. Innovative approaches are required to overcome the shortfalls of current systems and to design Next Generation Internet. In this context, an ambitious vision of future Inernet would include network virtualization which presents a viable solution to deal with the current Internet impasse. It provides a promising way to deploy different network architectures and protocols over a shared physical infrastructure. However, in spite of its multiple advantages, network virtualization adds more complexity on network systems. A promising solution to address this huge complexity consists in developing systems which are capable of managing themselves, called autonomic computing systems or self-* systems. This paper proposes an agent-based autonomic framework which is able to self manage virtual resources. We provide a detailed description of the proposed autonomic architecture and we focus on a real test-bed implementation and testing of our framework. Experiment results show the ability of our system to self-configure its resources in order to maintain a required QoS level.

Keywords: Network Virtualization, Autonomic Systems, Next Generation Internet, Network Architectures, Resource Management

1. INTRODUCTION

In the few last years, network virtualization concept has attracted a great deal of interest from both industry and research communities as an important enabler for designing the future Internet architecture. Internet's success stimulated the development and the deployment of new technologies and advances applications. However, the largest public wide network becomes victim of its own success. Its size and scope are now creating obstacles to future innovations and make difficult the introduction and the deployment of new network technologies [G. Schaffrath 2009][T. Anderson and Turner 2005][Turner 2005].

So, an ambitious vision of future Internet would include network virtualization. This new paradigm provides a promising way to run multiple architectures simultaneously on a single infrastructure. It enables the sharing of a physical network between many virtual networks and provides a clean separation of services and infrastructures. Besides, it facilitates new ways of doing business by allowing the trading of network resources among multiple providers and customers [Turner 2005][N. Feamster and Rexford 2007]. However, network virtualization adds more complexity on network systems. Indeed, every service is hosted inside a virtual machine which itself is hosted inside a physical equipment. These virtual and physical machines must communicate with each other in a reliable manner to guarantee user's requirements and needs. In

such a complex and dynamic environment, autonomic management approaches are needed.These approaches aim to address problems associated to current network management by pushing the responsibility of ensuring the proper operation of network to algorithms and processes that exhibit autonomic characteristics [Horn 2001].

In 2001, IBM proposes the "Autonomic Computing" paradigm [Research] to manage the complexity increase in the computing systems. Autonomic Computing is a system management referential that aims to introduce in the systems' core self-regulation mechanisms. The term "autonomic" comes from the human anatomy vocabulary, where the "autonomic nervous system" means the part of our nervous system whose role is the self-regulation of our organism. IBM Research [Horn 2001] has defined four properties namely self-configuration, self-optimization, self-protecting, self-healing known also as the self-* functions. They have suggested in addition to the self-* properties, a reference model for autonomic control loops necessary to achieve autonomic computing. The autonomic networking pursues the same objectives applying to the large-scale networks. Its goals are to overcome the network complexity by developing new kind of networks capable to self-manage and to support the upcoming growth and complexity.

We outline that the main contribution of our paper is to propose an autonomic framework which is able to self-provision and self-manage virtual resources. The goal of the proposed multi-agent based framework described in this paper is to address "cleverly" management's complexity and to offer reliability and scalability for virtualized networks.

The paper proceeds as follows. Section 2 describes the autonomic computing trend and its basic principles. In section 3, we present an overview of network virtualization. Section 4 summarizes the related work. In section 5, we propose and describe AAVP: an Autonomic Architecture for Virtual network Piloting to deal with instantiated resources during the lifetime of the Virtual Network. We provide a detailed description of the proposed AAVP architecture in Section 6. We describe the testbed set-up and experimental results in section 7. Finally, section 8 concludes the paper and presents our ongoing work.

2. AUTONOMIC COMPUTING

Autonomic computing paradigm is an emerging field for developing self-managing computing systems which are able to take care of their own behavior and their interactions with their environment without human intervention [Kephart and Chess. 2003]. This idea is essentially inspired from autonomic human body nervous system which, for example, monitors the heartbeat and maintains a healthy blood sugar level and a normal body temperature without any conscious effort from the human. This interesting concept applied to computing systems represents a promising solution to deal with the ever-increasing complexity of managing large scale IT systems.

The main four key self-management properties of autonomic computing as defined by IBM [D. F. Bantz and Vanover 2003][IBM] are:

—self-configuring: defines the ability of the system to configure and reconfigure automatically itself according to high level objectives in a changing environment.

—self-optimizing: defines the ability of the system to optimize efficiently the use of available resources.

—self-healing: defines the ability of the system to discover and repair potential problems to ensure that the system operates smoothly.

—self-protecting: defines the ability of the system to detect, identify and protect automatically itself against malicious attacks.

As a special case of autonomic computing, autonomic networking is concerned with creating self-directed and self-managed networks based on collected monitoring information and high level objectives given by administrators [S. Schmid and Hutchison 2006]. The network will be able to operate and serve automatically its purpose even in the case of environment changes. Emerging approaches to both management and control fields must be defined and used.

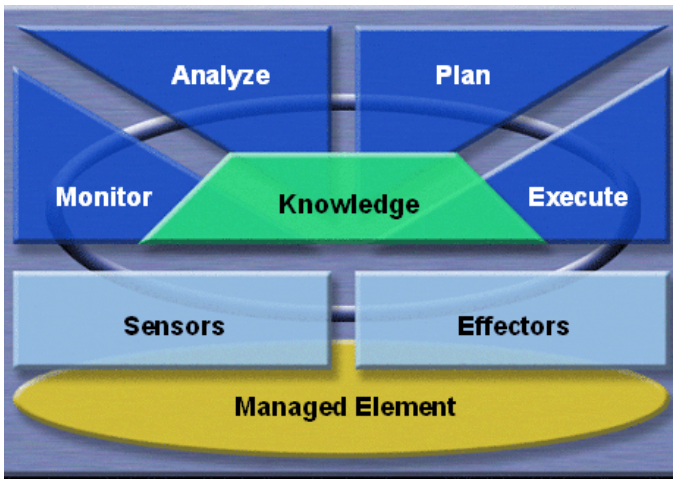


Figure. 1. IBM MAPE-K loop Reference Model

Since the IBM initiative for autonomic systems in 2001, many research and industrial projects have been carried out proposing different trends and frameworks in order to apply autonomic computing concepts to various areas of research [AuToI] [ANA] [Kephart and Das 2007] [SOCRATES]. They differ essentially by the distribution model of the proposed autonomic architecture (hierarchical or flat), the implementation of the MAPE-K loop, the interaction between the different autonomic elements and the used adaptation approach. Each proposed autonomic framework or architecture was designed to fit a specific use case and context of application and was focused on some self-managing properties.

We present in the following two key elements in the autonomic system processes: the MAPE-K loop reference model and network adaptation approaches defined in the literature.

2.1 MAPE-K Loop Reference Model

An autonomic network is composed of a set of autonomic elements which cooperate between each other in order to meet global and local predefined goals. As depicted in Figure 1, an autonomic element consists typically of one autonomic manager which manages automatically one or more managed elements [Research]. A managed element can be a hardware, software or network resource, a service or an application.

The autonomic manager can be configured by human administrators using high-level goals. It consists of:

—a *monitor*: which is responsible for knowledge gathering, collecting information about the managed element and capturing measurements of the environment.

—a *knowledge base* : where policies and monitored information is stored.

—an *analyze and plan* component: which analyses knowledge and plans appropriate actions.

—an *executor*: which reconfigures the system regarding the output of analyze and plan processes.

To achieve self-management tasks, the autonomic manager monitors knowledge from managed elements and its external environment thanks to sensors. It plans adaptation actions based on both the analysis of this knowledge and high-level directives predefined by the human administrator. Then, it executes the decided plans. Appropriate changes to the managed element are carried out by effectors. This process forms the autonomic control loop well-known as MAPE-K (Monitor, Analyze, Plan, Execute and Knowledge) model [McCann 2008]. Based on this autonomic control loop, the autonomic system adapts automatically its behavior in response to environment changes. This adaptation may be triggered periodically or in response to observed

external or internal events [Samaan and Karmouch 2009]. We present in the following, the main network adaptation approaches.

2.2 Network Adaptation Approaches

We distinguish in the literature two major adaptation approaches of autonomic networks: policy-based and utility-function strategies.

—In the case of *policy-based adaptation approach*, the adaptation plan is described by simple event-condition-action policy rules. Policies define the actions to be taken when an event occurs under specific conditions [Calo and Sloman 2003]. This strategy is simple to define. Besides, a typical advantage of policy-based autonomic networks is their capability to self-adapt at runtime by modifying the applied policies without stopping the network operation. However, this policy-based adaptation approach presents some limitations. In fact, it is impossible to the human operator to precisely predefine all reconfiguration policies in response to network environment changes that may occur. He has to predict all network condition changes, foresee all network events, describe desired behaviors and plan suitable actions, which is out of the human capacity. Omitting the corresponding policy of a particular network behavior may lead to a non optimized resource use or a crush in the system. Another main issue facing the usage of policies is the problem of policy conflicts and inconsistency of same taken decisions [Kamoda and Broda 2005]. For example, an event may satisfy the application conditions of two different policies presenting actions conflicts that could be detected only on runtime. Conflict resolution mechanisms should be defined at runtime. So, we note that in the case of the use of policy-based adaptation strategy, we can minimize the human intervention. But, the presence of the administrator is needed in the control loop to solve policy conflicts and unpredictable situations when they arise.

—On the other hand, the *utility-function adaptation approach* is based on abstract measures of possible configurations of network parameters [Palomar and Chiang]. Utility functions are basically used for self-optimization design patterns. The network resource allocation problem can be formulated as a constrained optimization (usually maximization) of some utility function [Kephart and Das]. The latter may be a function of throughput, packet delivery ratio, packet loss rate, delay and jitter. The autonomic manager based its reaction on the configuration which meets the higher network utility value. In comparison to policy-based adaptation approach, the use of utility functions fits better the optimal solution. However, the major drawback of utility functions is that human administrators find them difficult and awkward to specify [Kephart].

3. STATE OF THE ART AND OVERVIEW OF NETWORK VIRTUALIZATION

As depicted in Figure 2, a virtual network consists of a set of virtual nodes interconnected via dedicated virtual links. A substrate node is a physical equipment which is able to support many virtual nodes. Each virtual node belongs to a dedicated virtual network supporting a specific service or protocol. These virtual nodes are interconnected via virtual links shared over one or more substrate links.

The virtualization is not a new concept for the network community. IBM in the mid 1960s has developed the Virtualization specifically for mainframe computers. Currently, there are a number of virtualization techniques available, hardware and software approach, each of which addresses different issues. Virtualization software allows a single machine to run multiple independent guest operating systems concurrently, in separate virtual machines. These virtual machines provide the illusion of an isolated physical machine for each of the guest operating systems. A Virtual Machine Monitor (VMM) takes complete control of the physical machines hardware, and controls the virtual machines access to it.

Network virtualization presents a diversified Internet architecture. It supports multiple coexisting virtual heterogeneous networks, sharing a common physical substrate. Various architectures,

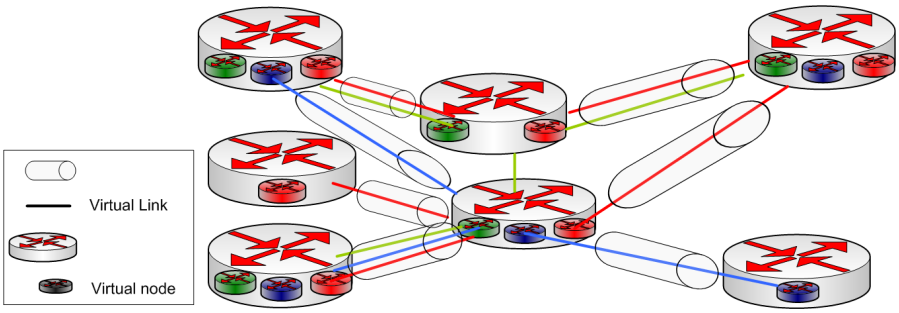


Figure. 2. Network virtualization model

experiments and services can be simultaneously supported by Virtual Networks (VN) [G. Schaffrath 2009] [T. Anderson and Turner 2005] [M. Yu and Chiang 2008]. In this section, we first present the concept of virtual networks. Then, we describe network virtualization model and actors. Finally, we illustrate the related business scenario.

3.1 Virtual network vs VPN

The concept of network virtualization is not new. The co-existence of multiple virtual networks appeared in the networking literature with the VPN (Virtual Private network) based networks. VPNs have known a great success of deployment last years. They connect multiple distributed sites of one or more companies through tunnels over shared or public networks such as the Internet [Andersson and Madsen 2005]. In comparison to VPN, actual network virtualization concept, has offered new commodities among others:

—Virtual networks sharing the same physical infrastructure may have different technologies and protocol stacks. This enables the coexistence of different networking solutions which is not possible with VPN.

—Virtual networks provide a real isolation of virtual network resources which is not possible with VPNs.

—The roles of the infrastructure provider and the virtual network provider are clearly separated in virtual networks. However, they are played by the same entity in the VPN.

3.2 Network virtualization' actors

Actors in the network virtualization model are different from those in the traditional networking model. The principal actor in the current Internet is the Internet Service Provider (ISP) [Ferguson and Huston 1998] [Rosen and Rekhter 1999] [Rosen and Rekhter 2006]. In network virtualization, this main actor is decoupled into two actors: Infrastructure Provider (InP) and Virtual Network Provider (VnP). From commercial point of view, this decoupling amortizes high fixed cost of maintaining a physical presence by sharing capital and operational expenditure across multiple infrastructure providers [Chowdhury and Boutaba 2010].

3.2.1 *Infrastructure Provider (InP).* The infrastructure provider owns and manages physical network resources in the network virtualization environment. It offers virtual resources to Virtual Network Providers who are its direct clients. It does not offer direct services to virtual network end users. We note that a physical infrastructure can be shared by many InPs. They communicate and collaborate among themselves to create the complete underlying physical infrastructure (4 Fig 3). They are also responsible for its maintenance.

3.2.2 *Virtual Network Provider(VNP).* The Virtual Network Provider is responsible for the creation and the deployment of virtual networks. It leases resources from one or multiple InP (3 Fig 3) to offer end-to-end services to its clients (virtual network users). It can also lease virtual

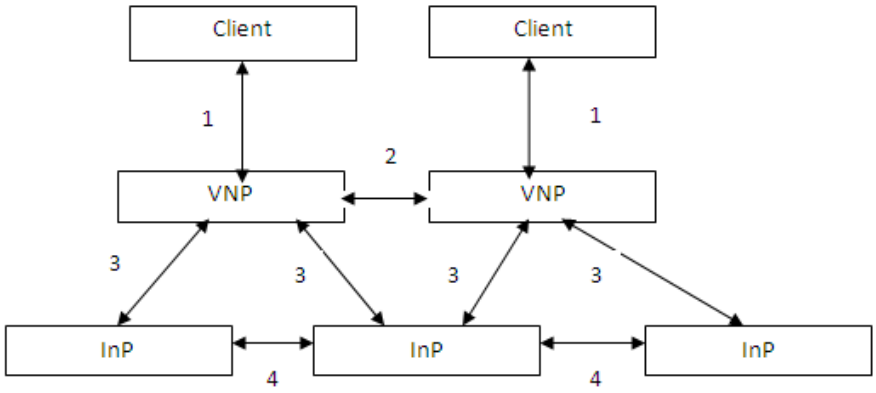


Figure. 3. Intraction model between network virtualization actors

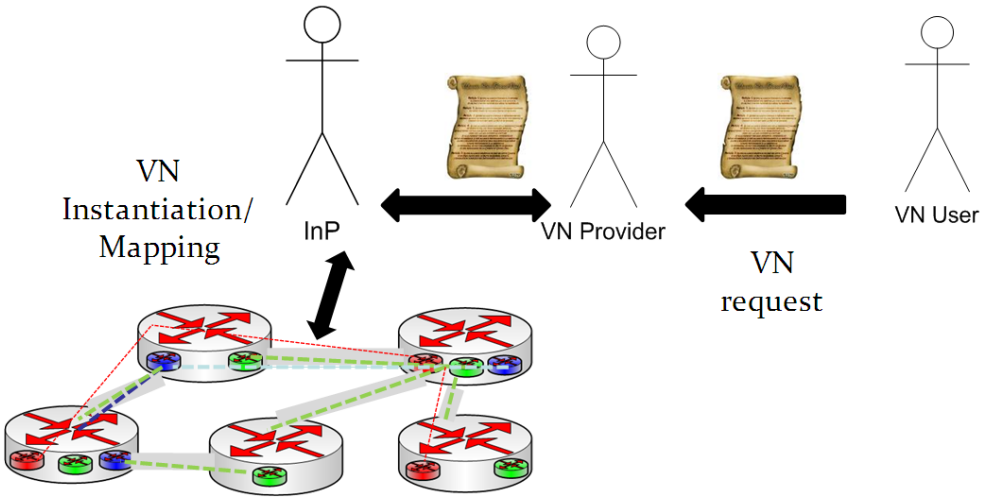


Figure. 4. Business Model

resources to Virtual Network Provider (2 Fig 3). We recall that each virtual network is managed by one VNP. A virtual network provider deploys its own protocols, services and applications in order to meet end user requirements.

3.2.3 *Virtual Network User (VN User).* The VN User requests a virtual network from Virtual Network Provider (1 Fig 3). A virtual network user may connect to multiple virtual network providers for different services. VN users actors generally correspond to end users, specific service providers (for example Internet Service Provider, video games provider, grid computing provider, etc), etc.

3.3 Business Scenario

As depicted in Figure 4, the client (VN User) specifies its service requirements. Then, it delegates the instantiation of the virtual network to the VNP of its choice.

Once the VNP receives the client service request, it generates a VN specification. Then, it negotiates the offer with candidate InPs and splits if necessary the VN resources description into multiple subsets. Based on the VN description, an InP identifies the appropriate substrate resources and allocates them. The InP has also the ability to migrate other VN in order to free

resources for new requests. Once the VN is instantiated, the client deploys its service.

So, in this context, network virtualization presents a promising concept for both industry and research community. However,in spite of its multiple advantages, network virtualization adds more complexity on network systems. A promising solution to address this huge complexity is the use of autonomic systems able to self-manage themselves. The next section describes the autonomic computing concept and its basic principles.

We present in the next section techniques defined in the literature to provision and manage virtual network resources.

4. RELATED WORK: VIRTUAL NETWORK RESOURCES PROVISION AND MANAGEMENT

We find in the literature a number of systems and techniques that have been put forward to provision and manage virtual networks resources.

4.1 Virtual Network Resources Provision

We summarize, in the following, the main techniques of virtual network resources provision presented in the literature. We give an overview of each proposal and we show, particulary, its drawbacks and limitations.

In [Zhu and Ammar], the authors proposed three virtual network(VN) embedding algorithms for virtual network assignments with dynamic reconfiguration, denoted by VNE-Least, VNE-Cluster and VNE-Subdividing. The first method, VNE-Least, treats virtual nodes and links mapping separately where substrate and virtual nodes are sorted according to their stress and connectivity degree respectively. Thereafter, VNE-Least makes use of the shortest distance algorithm to connect the mapped virtual nodes. On the other hand, the VNE-Cluster and VNE-Subdividing algorithms take into account the substrate link load when selecting the substrate nodes. Obtained results shows that proposed algorithms can achieve good performance in terms of load balancing. However, the authors did not consider real-life scenarios. They have assumed that the resources are unlimited in the substrate network (SN), which is not a realistic assumption.

In [M. Yu and Chiang 2008], the authors proposed the VNE-Greedy virtual network embedding algorithm. In this system, the substrate and virtual nodes are sorted according to the available and requested resources respectively. The main idea of the proposed embedding algorithm is the following. The virtual node with the highest resource request is assigned to the substrate node containing the largest available resource metric value recursively until all the virtual nodes are mapped.Simulations demonstrate that path splitting and path migration enable a substrate network to accept more virtual network requests. However, the main drawback of VNE-Greedy is the substrate path building algorithm. Indeed, the shortest path algorithm does not consider the congested SN links, which implies an increase of hot-spots in the substrate network and an increase in request reject rate.

The authors of [J. Lu 2006] model the VN as a directed graph with two types of nodes: access and core. The required VN resources are defined in terms of the expected traffic, which is expressed as an upper limit on allowed traffic between all access node pairs.The proposed method produces results that are close to a lower bound. Nonetheless, the weakness in the proposal consists in the lack of consideration of substrate network capacities (i.e. unlimited) and the use of static routing tables in the network. In addition, the proposal requires the star VN topology, which is strongly binding.

In [N. Chowdhury and Boutaba 2009], the authors propound two VN embedding algorithms, named Deterministic-ViNE and Randomized-ViNE. Here, the substrate graph is augmented with meta-nodes and meta-edges to form a meta-graph. Note that each virtual node is associated with a specified region where it could be hosted. Nevertheless, it is not realistic to expect end-users to specify all virtual nodes locations.Authors show that the proposed algorithms increase the acceptance ratio and the revenue of virtual Network requests while decreasing the cost incurred by the substrate network in the long run The main drawback here is the nodes locations constraints.

Indeed, when these are not defined, D-ViNE and R-ViNE cannot be executed since the meta-graph cannot be built.

We note that these proposed approaches are treated on a centralized way. Authors assume the existence of a central entity which has a global view of the entire network and all the information related to each node and link. Based on this vision, this centralized entity takes best Virtual network provision and configuration decisions. However, in a real environment, network parameters are very dynamic and equipments are numerous and heterogeneous. Hence, a central approach is not suitable and suffers from scalability limitations, information updates problems and high latency decisions.

4.2 Virtual Network Resources Management

To manage virtual networks, many papers present different primitives and mechanisms. [Y. Wang and Rexford 2008] proposes VROOM (Virtual Router on the Move), a primitive for virtual network management. It offers a free move of virtual resources (routers and links) from one physical equipment to another to simplify physical network-management tasks. Moreover, [Menasce and Bennani 2006] proposes techniques for dynamic allocation of processing resources (CPU) to virtual machines. Furthermore, in [P. Ruth and Goasguen 2006], authors present an autonomic system called VIOLIN. It is a virtual computational environment composed of virtual machines capable of live migration across a multi-domain physical infrastructure.

Besides, an autonomic approach for virtual resource control and management was proposed in [M. Kim Myung S. Kim and Hong 2005]. This approach provides a system based on autonomic computing and virtual networks concepts to meet SLA-based IP packet transport service 's requirements on core network infrastructures. However, this proposed architecture has not been implemented and no examples have been presented to instantiate different components. So real performances of proposed system are unknown.

In [M. Yu and Chiang 2008], the authors restrain the reconfiguration problem. In fact, migration is allowed to the virtual links and prohibited to the virtual nodes. To do so, first the reconfiguration algorithm periodically detects the over-loaded substrate links. Then, it finds new substrate paths or updates the split ratio of virtual links that are in transit within overloaded substrate links. Note that the authors base their proposal on traffic path splitting. Moreover, the authors do not take advantage of migrating traffic sources and sinks (i.e. virtual nodes) to minimize bottlenecks in the substrate network.

In [Marquezan et al. 2010], the authors propose an autonomic and distributed reconfiguration algorithm. It is run locally within all substrate nodes. The main idea is to shorten the physical path embedding a virtual link that overloads at least one substrate link according to its incoming/outgoing traffic. To do this, either the source or the destination of traffic (i.e. virtual node) is moved in order to shunt the overloaded substrate link. The reconfiguration algorithm is divided into five stages. First, each substrate node monitors and analyzes the presence of overloading traffic. Then, it exchanges monitoring information with its neighbors. Next, each substrate node analyzes the received information and decides whether to migrate one of its hosted virtual nodes or to receive a virtual node from its neighbors. After that, a receiving substrate node allocates the resources required to host the moving node. Finally, the virtual node is moved and hence the path length is reduced. The main criticism contracting a path may require a great deal of moving until the paths length becomes equal to one hop. Besides, the migration frequency of routers depends on the traffic load, which is actually unstable and correlated to the running applications.

5. AAVP:AN AUTONOMIC ARCHITECTURE FOR VIRTUAL NETWORK PILOTING

In this section, we propose an autonomic system to provide resources and manage virtual networks. Using high level goals and based on distributed algorithms and network level knowledge, autonomic entities making our system collaborate together to instantiate and manage virtual resources. This minimizes human intervention and leads to an effective cost operator

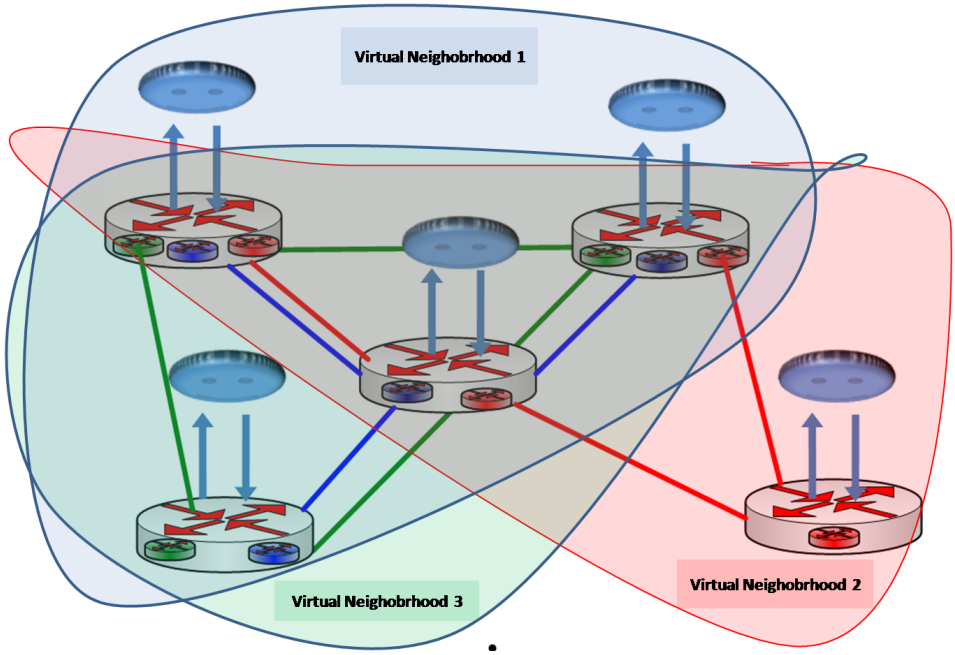


Figure. 5. Virtualized Network Infrastructure

Our autonomic and distributed system aims essentially to:

—Reconfigure its instantiated virtual networks at run time as network conditions change over time due to the arrival and departure of VNs,

—Optimize the use of its resources whether physical or virtual to maximize its service revenue. It could be through forecasting variations and future demand,

—Localize, diagnose and identify the problem then repair it by itself and without human intervention.

5.1 Network Infrastructure

As depicted in Figure 5, our autonomic system is composed of physical network equipments (routers, access points, etc.) interconnected with each other through physical links. Each network equipment is able to embed many virtual nodes. Virtual nodes are interconnected with each other through virtual links embedded in physical links. These physical network equipments are piloted by autonomic entities. In order to pilot virtual resources, autonomic entities exchange knowledge within the range of a logical and physical neighborhood. The knowledge concerning the neighborhood of each autonomic entity is called "situated view".

5.2 Situated View

Autonomic entities (i.e agents) communicate periodically with each other through their situated views. The Situated View of each agent is a structured knowledge base representing the environment of the agent. It contains knowledge elements collected locally by the agent as well as knowledge obtained from its peers.

The Situated view represents the environment vision of an autonomic entity. This environment is the knowledge concerning local equipment and its neighbors. As depicted in Figure 5, an autonomic entity may have two types of neighbors:

—Physical neighbors which are neighbors that are physically connected to autonomic entities,

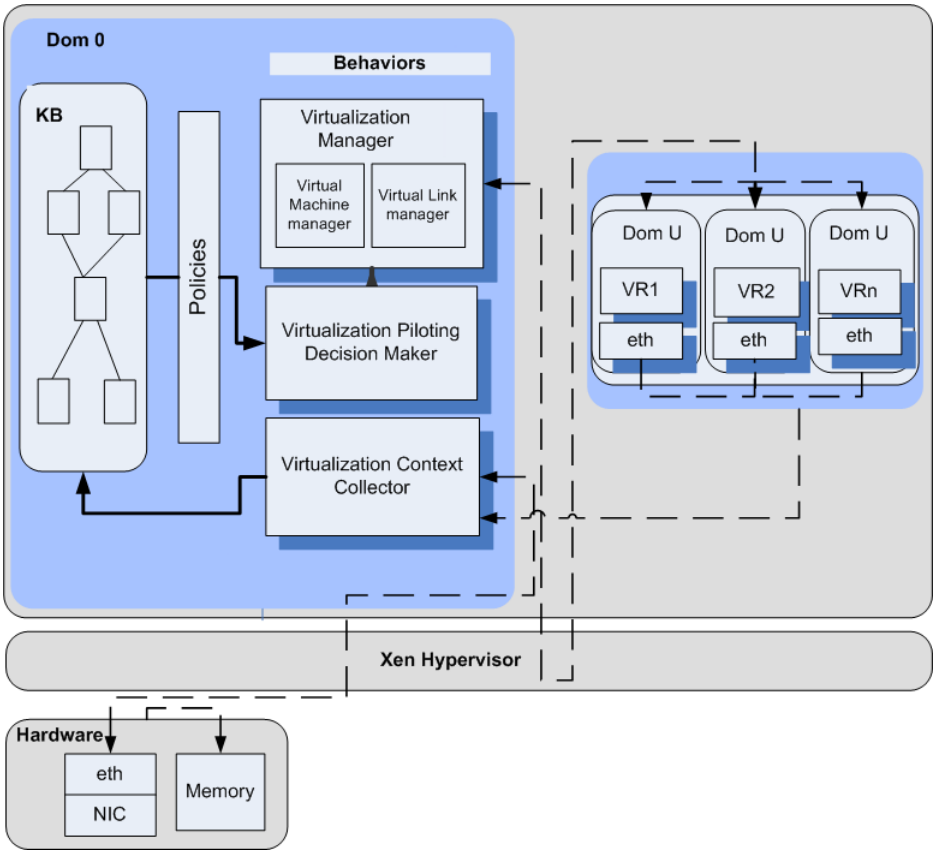


Figure. 6. Autonomic Piloting Virtual Network Architecture

—Logical neighbors which are virtual neighbors. An autonomic entity maintains a neighborhood for each virtual network. Indeed, a virtual network is composed of a set of virtual nodes embedded in physical nodes. Two neighbors of a virtual network can be physically distant which means that are not physically connected. So, each physical node maintains a neighborhood for each virtual node mapped on it.

The Situated View is updated on a periodical basis and it is used to adapt the Behaviours to changes occurring in the network in order to take real-time decisions. An automatic mechanism mirrors the Situated View to the appropriate peers depending on the type of neighborhood (physical or virtual). The knowledge is reflected in the Situated View of the peer agents. The rate and range of this mechanism can be tuned according to the nature of the knowledge. The Situated View is organized following an ontology-based model which is detailed in section 5.3.1.

5.3 AAVP description

We propose in this article, an autonomic agent-based platform to manage the complexity of virtual network. Moreover, our designed platform is proposed for large scale network. It is distributed and this distribution is possible thanks to autonomous agents which are embedded in routers and disseminated over the network.
An agent is a piece of software which is able to evolve in an uncertain environment and possesses the autonomy to make decisions.
As shown in Figure 6, our architecture consists of the following components:

5.3.1 *Knowledge Base (KB).* The knowledge base represents the core of our autonomic architecture. It offers a common vocabulary to different network equipments which may have different data management tools. Thanks to the knowledge stored in its KB, each autonomic entity acquires a vision of its own equipment and its environment. The knowledge base is organized of classes connected to each other in order to describe the virtual environment of an equipment. These classes are instantiated on individuals which are regularly diffused in a predefined neighborhood (one neighborhood is defined by either a shared network medium or a list of Piloting Agents). New individuals are automatically added to the Knowledge Base of a Piloting Agent upon their receipt from another agent. The situated view concept described above is implemented thanks to the knowledge base.

Figure 7 depicts the knowledge base model which is represented in UML. As shown in Figure 7, a topology is a set of nodes interconnected with each other through links. A node embeds an agent and a device. The former represents the proposed autonomic architecture. The latter describes the network equipment structure. In fact, a device may be either a physical device or a virtual one and has static parameters such as location (Region, city, etc.), operating system (Linux, Windows, etc.), virtual machine nature (Router, Switch, Access Point, etc.), virtual machine environment type (Xen, VMware, etc.), network stack type (MPLS, TCP/UP, etc.) and interface type (Ethernet, Atm, radio, etc.). A physical device may embed one or more virtual devices and has functional attributes such as available CPU, available memory and available storage. A virtual equipment has also functional attributes such as CPU usage level, memory usage level and storage usage level. Devices are interconnected through links which have their own metrics such as bandwidth, loss rate, end to end delay, etc. These metrics are computed through raw information gathered from network interfaces (e.g. sent packets rate, received packets rate, lost packets rate, etc.).
In our implementation, each router may embed a wireless card. So, a router may play the role of an access point or a base station depending on used technology (WIFI, WIMAX, UMTS, etc.).

5.3.2 *Policies.* Policies define rules that control the triggering of behaviors according to the current state information and context.
Policies are defined by the network infrastructure operator in order to meet the customers SLA requirements in terms of resources and QoS. They may be updated in function of changes of network environment and users.
We proposed in [I. Fajjari and Pujolle 2010] a virtual resources provisioning schema that specifies virtual resources proprieties and associations. This schema is used to instantiate new virtual networks in function of user requirements and the contract on which it agrees with its operator.

5.3.3 *Behaviors.* Behaviors can be viewed as organic components permanently sensing the environment and acting upon it. Technically, behaviors are specific functions executing precise tasks for specific goals. To accomplish their tasks efficiently, they use knowledge information stored in the knowledge base.
We define five principal behaviors:

—**Virtualization Context Collector (VCC)**: This behavior is responsible for supervising and monitoring physical and virtual resources within the physical node. Thanks to the interface Autonomic entity/Network equipment, autonomic entity retrieves raw information and generates metrics that describe the network equipment state. Theses metrics are stored in the knowledge base which is periodically updated.

—**Virtualization Piloting Decision Maker (VPDM)**: This behavior makes decisions according to the knowledge stored in the knowledge base. VPDM decision making is based on the execution of management and instantiation algorithms which must be previously designed. Decision maker can order the instantiation or the delete of new virtual router. It can also order the tuning of the amount of virtual resources allocated to each virtual router. The deci-

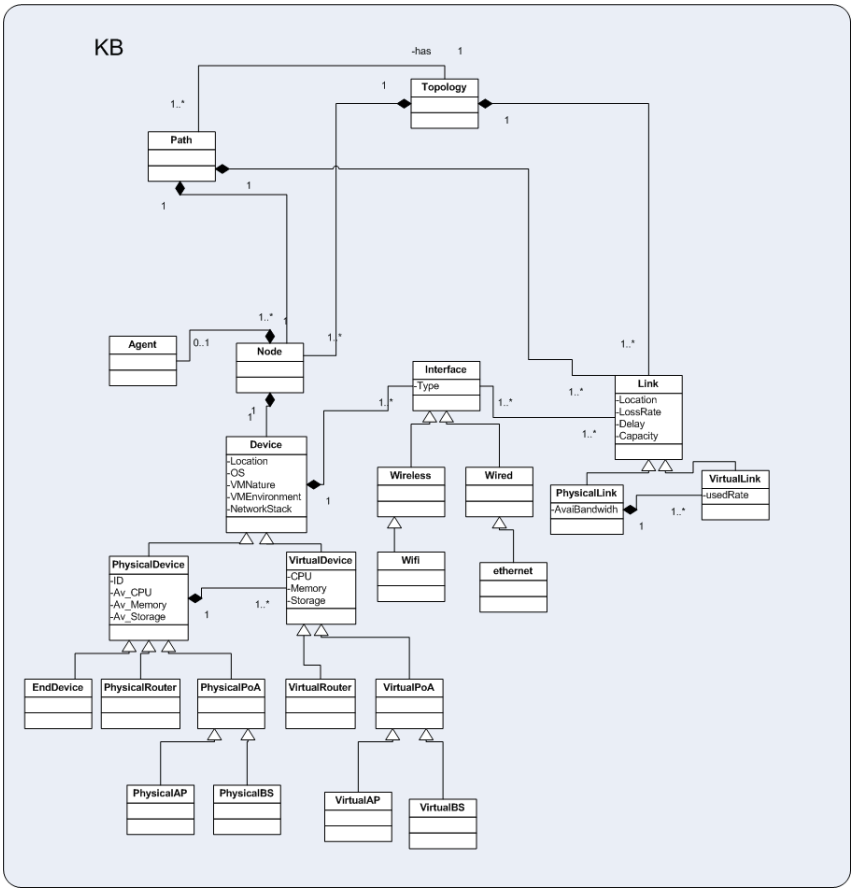


Figure. 7. Ontology-Based Model for virtual network knowledge base

sion made depends essentially on current state of physical network equipment, the state of the network and the SLA fixed for each type of virtual network.

—**Virtual Resources Managers (VRM)**: These behaviors are in charge of executing actions upon virtual resources according to the decision taken by the behavior "Virtualization Piloting Decision Maker". We distinguish:

—**Virtual Machine Manager (VMM)**: This behavior manages virtual nodes instantiated in the physical network equipment. Management tasks may be: "instantiate" a virtual machine which means creating a new instance of a virtual machine according to a specific specification, "migrate" a virtual machine that means change the instance of a machine virtual from the local network equipment to a foreign network equipment due to a lack of resources, "destroy" a virtual machine that means deleting the virtual machine and its bookkeeping information, "suspend" a virtual machine which means pause a virtual machine and store its internal state on a file disk, "stop" a virtual machine, "resume" a virtual machine that means executing a virtual machine from a state saved on a file disk.

—**Virtual link Manager (VLM)**: This behavior manages virtual link instantiated. Task management may be "instantiate" a link, "remove" a link, "modify" a link which means tuning link parameters of a virtual link, and migrate link.

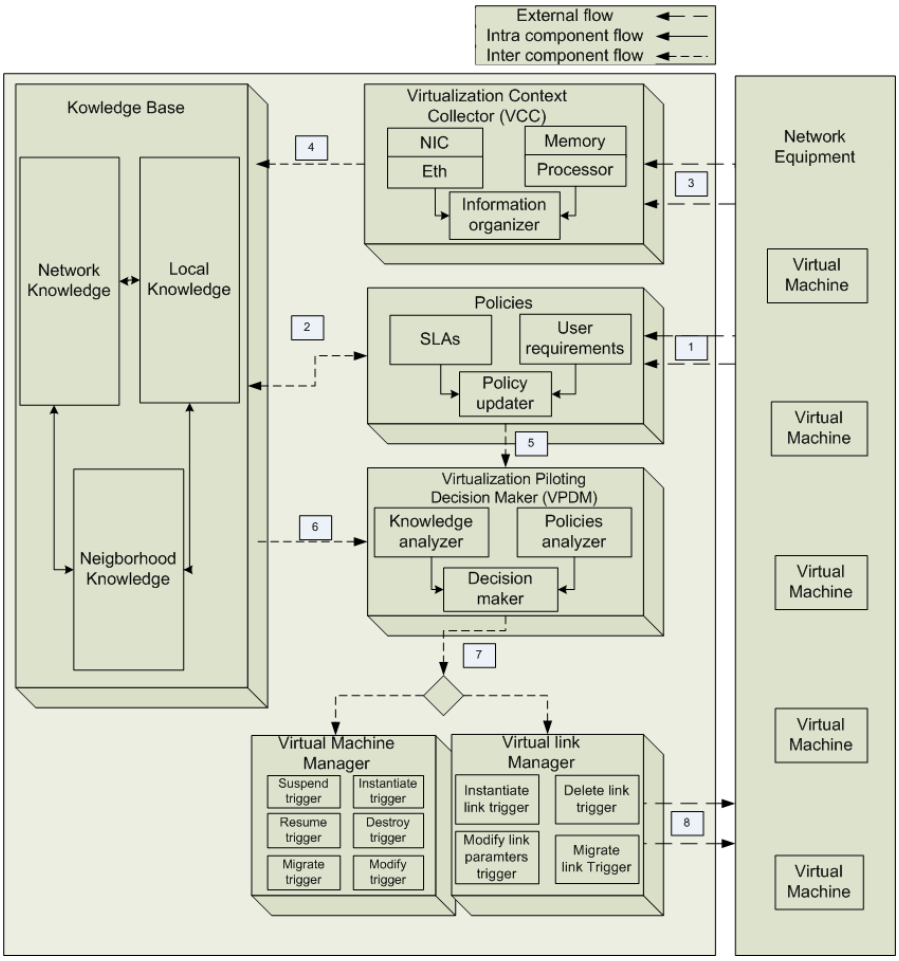


Figure. 8. Detailed AAVP architecture

6. AAVP: DETAILED DESCRIPTION

We provide in this section a detailed description of the implemented architecture. We note that we use policies as a set of rules to manage and control the access to network resources. They control the triggering of behaviors according to the current state information and context.
In our model, a policy is essentially a set of event-condition-action rules. Rules are organized hierarchically in sets called sub-goals. Each sub-goal corresponds to a set of rules.

During a policies evaluation cycle, rules with matching condition are triggered. Once a rule is triggered, the list of primitives in the right part of the rule is executed.
We proposed in [I. Fajjari and Pujolle 2010] a VN-SLA that defines the policies between different actors in terms of guaranties and penalties.
Figure 8 illustrates the detailed building blocks of AAVP architecture and the interactions between them.
Policies are defined by the network infrastructure operator in order to meet the customers SLA requirements in terms of resources and QoS and operator goals (1,2 Fig. 8).

To meet changing context, Policy Updater aims to adapt the policies according to user requirements, environment conditions and operator goals. This process may be automatic thanks to meta-policies. It may also be manual. Sometimes, the network infrastructure operator intervention may be required in order to update and refine predefined policies.

As we have mentioned in the previous section, the VCC implements the information organizer component which is responsible for the raw information structuring.
This component updates continuously the knowledge base according to the changing context of the virtualized network. VCC collects raw information from network equipment(3 Fig. 8) thanks to existing tools such as Xentop. Then, the information organizer subcomponent analyses, aggregates and generates new metrics related to virtualized resources (e.g VCC generates the metric CPU usage level of a virtual machine according to the number of CPU units used by the latter during a time interval T). These metrics as CPU usage, memory usage, error rate etc. are stored in the knowledge base (4 Fig. 8).

VPDM is an intelligent component which acts on behalf of administrators and users and manages virtual resources in an autonomic way. It aims to maintain a desired level of QoS for each instance of virtual network. VPDM analyses continuously policies (5 Fig. 8) and knowledge stored in KB (6 Fig. 8). In the case of context information change, it uses heuristics to optimize the management of virtualized resources. For example, if a physical machine can no more support all instantiated machines above it due to a lack of resources or a network bottleneck, it tries first to tune allocated resources of one or more virtual machines according to the SLA of each one. If resources shrinking is not possible or not competent, the physical machine will attempt to search within its physical situated view a physical node to which it can migrate one or more virtual machines. Other mechanisms of optimization can be proposed to maintain the efficiency of the substrate topology. Each physical machine fixes a desired level of allocated resources and tries to maintain continuously a load balancing with others physical machines. In fact, when physical machine becomes overloaded, it triggers a process of migration to an under-utilized physical machine. Moreover, power saving mechanisms can be used to reduce the power consumption in routers, such as turning off under-utilized physical routers and migrating their virtual resources to other physical routers able to host more virtual machines.

All decisions made by VPDM are communicated to VMM and VLM (7 Fig. 8) which triggers appropriate actions on the network equipment (8 Fig. 8).
Then, VCC updates the knowledge base according to the changing context of the virtualized resources.
As depicted in Figure 8 our architecture is knowledge centred. In fact, most components interacts with the knowledge base in order to guarantee their functionalities.
Unlike, [Zhu and Ammar][M. Yu and Chiang 2008][J. Lu 2006] and [N. Chowdhury and Boutaba 2009], our solution is scalable thanks to its distributed nature, our Multi-agent System scales well with the growing size of the network. For that, an autonomic agent is integrated on every node to be controlled. The control (configuring, monitoring) of these nodes is realized in a scalable and autonomic way thanks to the situated view. the autonomic node controls only local parameters. However, the autonomic node can use information from the knowledge plane to adapt the control process. However, the distribution of the physical resources could be a very difficult problem. For this reason, an autonomic piloting system was designed to take in charge this control. The piloting system is based on a piloting plane which is handled within the hypervisor. Moreover, unlink [M. Yu and Chiang 2008], our system takes advantages from nodes and path migrations which offers more flexibility and efficiency to the reconfigured network. Besides, each autonomic updates its local information stored on its knowledge base only when this latter is changed which reduces useless updates like in [Marquezan et al. 2010]. We have implemented our autonomic architecture in java using Xen environment. In Xen, each virtual machine is hosted in a Guest domain called DomU. Among Guest Domains, there is a single domain which is able to access directly to physical resources. It is called Dom 0.
Our autonomic architecture as presented in Figure 6 is hosted in Dom 0. This domain is responsible for resources sharing such as CPU and memory. It controls the execution of different

virtual machines inside the physical network equipment. We describe in the following section our testbed and preliminary implementation results.

7. EXPERIMENTAL STUDY

In order to have an overview of the the effectiveness of our architecture and its ability to self-configure its resources under specific scenarios, we have chosen to set-up a real-world testbed instead of network simulations. In fact, due to implementation short cuts and the simplification of some real-world properties, simulation techniques may lead to results and conclusions which do not reflect the behavior of our solution under realistic constraints.

7.1 Experimental Setup

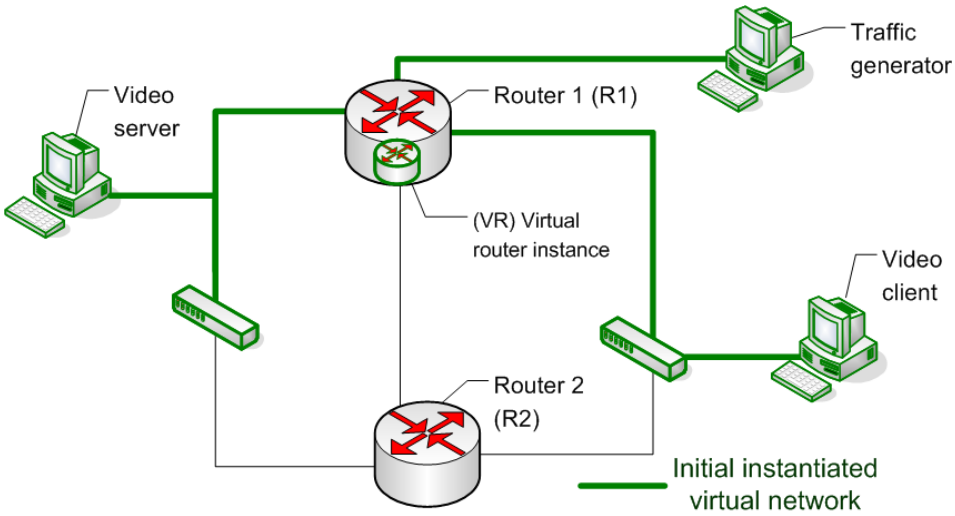


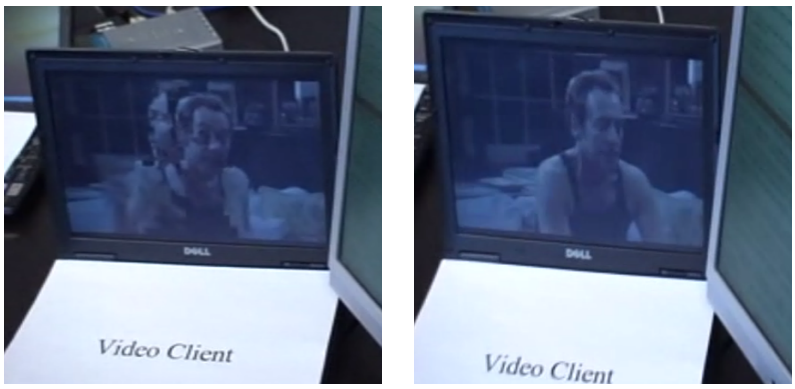
Figure. 9. Virtual Network Testbed

As depicted in Figure 9, our experimental testbed consists of 2 Physical Nodes. The Physical Node has 4Go RAM, C2D-2.4 GHz CPU, six 1 GByte network interface. A Virtual Router VR1 has this configuration: one x86 virtual CPU, two 100 MB virtual interfaces, 20 Mb image disk size, 80Mb RAM and Quagga IPv6 router as network application. A virtual network VN1 is instantiated between a video server and a video client. Each one has C2D 1.6 GHz CPU, 2Go RAM and 100Mb network interface. VN1 passes through Router1(R1) and it is marked with green color in Figure 9. The video server sends a video flow with 1 Mbit/s to the video client. This video flow is displayed continually on the latter's screen.

We have performed a set of experimentations to check our autonomic agent ability on detecting network interfaces congestion and network performance degradation. We check also its capacity to make the appropriate decision in order to overcome detected problems and improve network performances. The main idea behind these experimentations is to show the reactivity of the proposed system and put forward the utility of the autonomicity. In fact, thanks to autonomic entities and the monitoring of the stored information in the knowledge base, the network is able to detect a possible deterioration. Then, it acts in order to solve the problem according predefined rules.

7.2 Scenario and Results

We considered the following scenario. At t=40s, a huge data flow is generated by the host Traffic generator directly connected to R1 and circulated in the virtual network. Due to this traffic,



(a) Video before migration (b) Video after migration

Figure. 10. Quality of Video

the video displayed in client video 's screen is getting fuzzy (Fig. 10 (a)) which confirms the performance collapse of the virtual network.

To trigger migration, we define the rule R as follows:

R: If PacketLossLevel ≥ Threshold Then trigger migration to the neighbor router having less loaded network interfaces.

We defined PacketLossLevel as:

$$PacketLossLevel = \frac{NbrLostSentPackets}{TimeIntervall} \quad (1)$$

where TimeIntervall=100ms

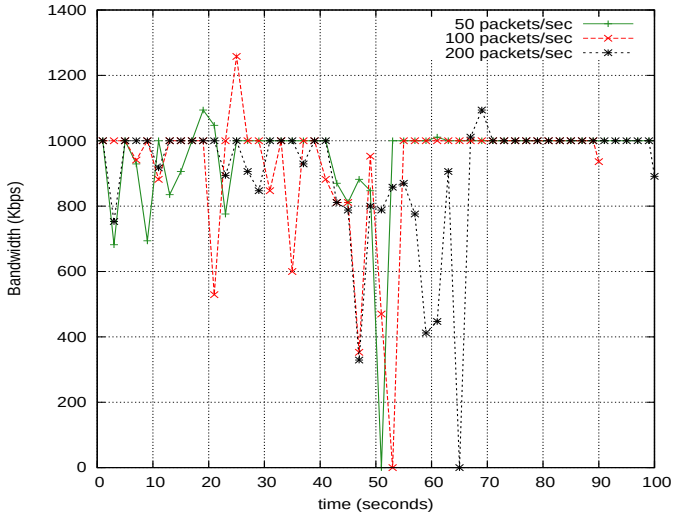
As soon as the rule becomes true, AAVP agent decides to migrate the virtual router instance (VR) which is embedded in router R1. Thus, it searches on its knowledge base the least loaded router which corresponds in this case to the router R2. Then, it triggers the move of VR to router R2. Thanks to this reconfiguration VN1 is able to maintain its performance and the required QoS which results in acceptance video performance as shown in Figure 10 (b). Without adaptation, the VN1 would have probably crashed due to the lack of available resources.

Firstly, we evaluated the performances of AAVP when varying the threshold. Figure 11 shows the bandwidth and the loss variations for each fixed threshold (50, 100 and 200 packets/sec). It is clear that the more the value of the threshold increases the more migration is delayed and the more the flow is disrupted. This is due to the huge flow sent to the physical router which deteriorates the router's performances. In fact, we notice that for a high value of threshold, the flow takes more time to reach its required QoS after the migration. For these reasons we fix the error threshold to 50 packets/sec.

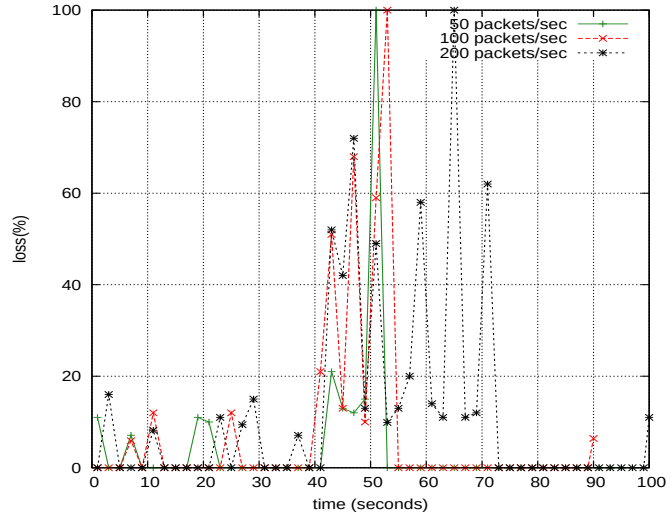
Experimentations shown in Figure 11 aim to find the best value of the error threshold defined in the rule (1). A best choice of this value will optimize the system and will lead to a fast detection of bottlenecked links thanks to packets loss evaluation. Figures 11 shows that fixing the error threshold to 50 packets/sec leads to good performances. In fact, the system triggers migration at the best moment and the flow takes accordingly a short delay to reach its required QoS after the migration.

Figure 12 displays bandwidth and packets loss variation throughout the scenario's execution. We note that, at the beginning, the bandwidth and the loss rate are not stationary. This is due to resources limitation. In fact, for R1, we fixed a low bound of maximum allocated CPU to VR1 in order to cause performance deterioration as soon as the heavy traffic is generated. This leads to a small perturbation of the flow circulating through VR network interface.

Figure 12 shows that when using our architecture, AAVP agent reacts to perturbation in less



(a) Bandwidth variation



(b) PER variation

Figure. 11. Network performances with varying trigger migration threshold

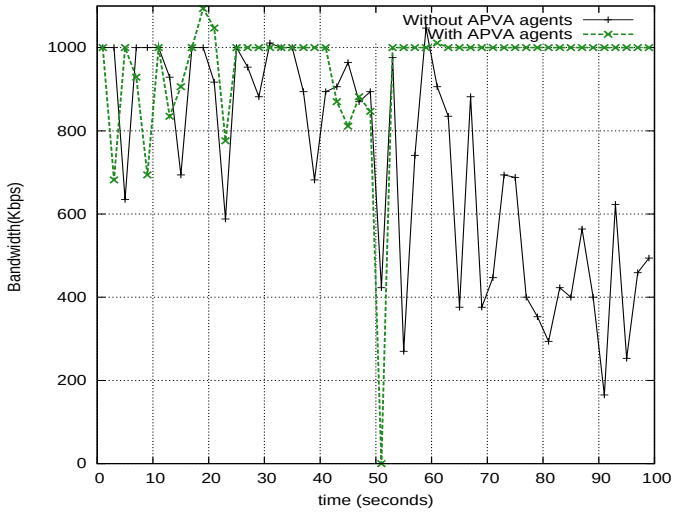
than 7s. In fact, AAVP agent, decides to trigger migration only when PacketLossLevel exceeds 50 packet/s which corresponds to a noticeable degradation of the video quality. In order to reduce reactivity duration, packetLossLevel threshold should be reduced. However this can lead to a premature migration in the case of brief perturbation.

Without a piloting architecture, it is clear that the video streaming server throughput requirement is no more respected due to the network's overload. This is clear through the continuous increase of the Loss Rate and the decrease of the Bandwidth.

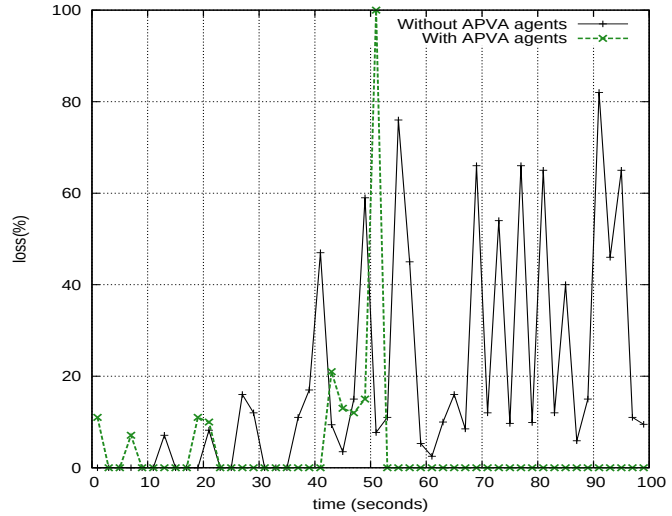
As depicted in Figure 12, the delay of migration, which represents the interruption time on the execution of the video running inside of the moving virtual slice, is less than 2s.

Moreover, results show that once migration is executed, the bandwidth becomes stationary (respectively loss becomes null) which guarantees the QoS required for the video stream.

Previous experiments demonstrate the effectiveness of our architecture and its ability to self-configure its resources in order to maintain a required QoS.



(a) Bandwidth variation



(b) PER variation

Figure. 12. Network performances evaluation with and without AAVP agents

8. CONCLUSION

Network virtualization is a promising technique to overcome the internet ossification by providing a shared physical infrastructure for a variety of network services and architectures. However, in spite of its multiple advantages, network virtualization adds more complexity on network systems. In order to address this complexity, we propose in this paper an autonomic architecture for virtual network piloting: AAVP. Each AAVP node consists of three main entities reflecting its ability to monitor, analyse and then manage and optimize the use of network resources.

We have implemented the described autonomic system using Xen environment. Real experimentations presented in this paper are satisfying and prove the ability of our system to automatically reconfigure itself and improve its performances. We are defining an utility function to model agents behavior in a situation of choice and migration decision. This function calculates the utility of each router according to its performances. We plan to evaluate the effectiveness of the proposed utility function and the performances of our system with large-scale experiments through simulations.

REFERENCES

ANA. Autonomic network architecture, fp7 project. In *[Online] Available: www.ana-project.org/.*

Andersson, L. and Madsen, T. 2005. Provider provisioned virtual private network (vpn) terminology. *RFC 4026*.

AuToI. Autonomic internet, fp7 project. In *[Online] Available:http://ist-autoi.eu/.*

Calo, S. and Sloman, M. 2003. Guest editorial: Policy-based management of networks and services. *Journal of Network and Systems Management* 3, 249–252.

Chowdhury, N. M. M. K. and Boutaba, R. 2010. A survey of network virtualization. *Computer Networks 54*, 862–876.

D. F. Bantz, C. Bisdikian, D. C. J. P. K. S. A. M. D. G. S. and Vanover, M. 2003. Autonomic personal computing. *IBM Systems 42,* 1, 165–176.

Ferguson, P. and Huston, G. 1998. What is a vpn? *Cisco Systems,Technical Report*.

G. Schaffrath, C. Werle, P. P. A. F. R. B. A. A. W. M. K. O. M. L. M. 2009. Network virtualization architecture: proposal and initial prototype. *ACM workshop on Virtualized infrastructure systems and architectures, VISA'09*, 63–72.

Horn, P. 2001. Autonomic computing: Ibm's perspective on the state of information technology, also known as ibm's autonomic computing. *Computing Systems*.

I. Fajjari, M. A. and Pujolle, G. 2010. Vn-sla: A virtual network specification schema for virtual network provisioning. *IEEE International Conference On Networks, ICN*, 337–342.

J. Lu, J. T. 2006. Efficient mapping of virtual networks onto a shared substrate. *Technical Report WUCSE-2006-35*.

Kamoda, H. and Broda, K. 2005. Policy conflict analysis using free variable tableaux for access control in web services environments. In *Policy Management for the Web Workshop*. 5–12.

Kephart, J. O. Research challenges of autonomic computing. In *international conference on Software engineering, ICSE'05*.

Kephart, J. O. and Chess., D. M. 2003. The vision of autonomic computing. *Computer Networks* 1.

Kephart, J. O. and Das, R. Achieving self-management via utility functions. *IEEE Internet Computing 11,* 1, 40.

Kephart, J. O. and Das, R. 2007. Achieving self-management via utility functions. *IEEE Internet Computing 11,* 1, 40–48.

M. Kim Myung S. Kim, A. Tizghadam, A. L.-G. and Hong, J. W.-K. 2005. Virtual network based autonomic network resource control and management system. *IEEE Global Communications Conference, Globecom*, 1075–1079.

M. Yu, Y. Yi, J. R. and Chiang, M. 2008. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review 38,* 2, 17–29.

Marquezan, C. C., Granville, L. Z., Nunzi, G., and Brunner, M. 2010. Distributed autonomic resource management for network virtualization. *IEEE Network Operations and Management Symposium, NOMS*, 463–470.

McCann, J. A. 2008. A survey of autonomic computing - degrees, models and applications. *ACM Computing Survey 40,* 3.

Menasce, D. and Bennani, M. 2006. Autonomic virtualized environments. *International Conference on Autonomic and Autonomous Systems, ICAS*, 28.

N. Chowdhury, M. R. R. and Boutaba, R. 2009. Virtual network embedding with coordinated node and link mapping. In *IEEE Conference on Computer Communications, INFOCOM'09*. 783–791.

N. Feamster, L. G. and Rexford, J. 2007. How to lease the internet in your spare time. *ACM SIGCOMM Computer Communication Review* 37, 61–64.

P. Ruth, J. Dongyan Xu, R. K. and Goasguen, S. 2006. Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. *IEEE International Conference on Autonomic Computing, ICAC*, 1–5.

Palomar, D. P. and Chiang, M. A tutorial on decomposition methods for network utility maximization. *IEEE Journal On Selected Areas In Communications 24,* 8.

Research, I. Ibm and autonomic computing: an architectural blueprint for autonomic computing. IBM Publication.

Rosen, E. and Rekhter, Y. 1999. Bgp mpls vpns. *RFC 2547*.

Rosen, E. and Rekhter, Y. 2006. Bgp mpls ip virtual private networks (vpns). *RFC 4364*.

S. Schmid, M. S. and Hutchison, D. 2006. Towards autonomic networks. *Autonomic Networking*, 1–11.

Samaan, N. and Karmouch, A. 2009. Towards autonomic network management: an analysis of current and future research directions. *Communications Surveys and Tutorials 11,* 3, 22–36.

SOCRATES. Self-optimisation and self-configuration in wireless networks, fp7 project. In *[Online] Available: http://www.fp7-socrates.org/.*

T. Anderson, L. Peterson, S. S. and Turner, J. 2005. Overcoming the internet impasse through virtualization. *IEEE Computer Magazine* 38, 34–41.

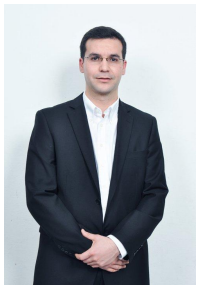
Turner, J. 2005. Diversifying the internet. *IEEE Global Telecommunications Conference, GLOBECOM*.

Y. Wang, E. Keller, B. B. J. M. and Rexford, J. 2008. Virtual routers on the move: Live router migration as a network-management primitive. *IEEE conference of the Special Interest Group on Data Communication, SIGCOMM 38.*

Zhu, Y. and Ammar, M. Algorithms for assigning substrate network resources to virtual network components. *IEEE Conference on Computer Communications, INFOCOM'06*, 1–12.

,

Ilhem Fajjari is currently a Ph.D. student in Phare team at the LIP6 laboratory Pierre & Marie Curie University (Paris 6) in France. She works under the supervision of Pr Guy Pujolle. She worked as R&D Engineer in Ginkgo Networks startup from 2008 to 2011. She received a M.S degree in Computer Science - Networking with honors from Ecole Nationale des Sciences de l'Informatique (ENSI) in Tunisia, 2009. She gained Engineer Diploma in computer science with honors from Ecole Nationale des Sciences de l'Informatique (ENSI) in Tunisia, 2008. Her research interests include network virtualization, autonomic computing and resources allocation in wired networks.



Othmen Braham was born in 03/09/1982 in a lovely tourist city Mahdia in Tunisia. He is currently leading research and development in VirtuOR. He obtained his computer science engineer from the ENSI (2007, Tunisia). He received his Master's degree in computer science from Paris VI (2008, France). Actually, he is accomplishing his PhD between LIP6 and VirtuOR. His research has been mainly devoted to the area of computer networks virtualization and Future Networks. Othmen Braham is actively involved in VirtuOR products development and providing solutions for virtualized environment. VirtuOR (www.virtuor.fr) is a very challenging company which offers a solution which guarantees services Cloud access to their customers.

Mouna Ayari received her PhD in Computer Science conjointly from ENSI, University of Manouba, Tunisia and University of Paris 6, UPMC - Paris Universitas, France in 2009, her M.S in network and computer science and her B.S. degree in computer science and network engineering from ENSI, Tunisia, in 2004 and 2003 respectively. Her research interests include autonomic networking, policy-based networking, femtocells, network virtualization and quality of service management in wireless and mobile networks. She is currently an assistant professor of computer science at ISIM - University of Monastir - Tunisia and member of the RAMSIS research team of CRISTAL Laboratory at ENSI, Tunisia.

Guy Pujolle received the Ph.D. and "Thse d'Etat" degrees in Computer Science from the University of Paris IX and Paris XI on 1975 and 1978 respectively. He is currently a Professor at the Pierre et Marie Curie University (Paris 6) and a member of the Scientific Advisory Board of Orange/France Telecom Group. He was appointed by the Education Ministry to found the Department of Computer Science at the University of Versailles, where he spent the period 1994-2000 as Professor and Head. He was Head of the MASI Laboratory (University Pierre et Marie Curie - Paris 6), 1981-1993, Professor at ENST (Ecole Nationale Suprieure des Tlcommunications), 1979-1981, and member of the scientific staff of INRIA (Institut National de la Recherche en Informatique et Automatique), 1974-1979. He is currently an editor for International Journal of Network Management, WINET, Annals of Telecommunications, and IEEE Surveys & Tutorials.

Hubert Zimmermann is currently a director at "Boost your startup" company. He was Director High Availability SW Engineering at Sun Microsystems in California. Hubert was also founder and CEO of Chorus Systems, a global system software company that he sold to Sun Microsystems. Prior to Chorus, Hubert led a number of R&D projects at INRIA, at France Telecom R&D and with the French Ministry of Defense. One of Hubert's other credentials was leading the development and the international standardization of the 7-layer OSI reference model at ISO and CCITT.