Towards an Autonomic Piloting Virtual Network Architecture

Ilhem Fajjari, Mouna Ayari, Othmen Braham, Guy Pujolle, Hubert Zimmermann

Towards an Autonomic Piloting Virtual Network Architecture

Ilhem FAJJARI*†, Mouna AYARI*‡, Othmane Braham§, Guy PUJOLLE* and Hubert ZIMMERMANN†

*LIP6 Laboratory, University of Paris VI, Paris, France
†Ginkgo Networks Company, Montrouge, France
‡CRISTAL lab, National School of Computer Sciences, University of Manouba, Tunisia
§VirtuOR, 4, Residence de Galande 92320 Chatillon, France
Emails: {Ilhem.fajjari, mouna.ayari, guy.pujolle}@lip6.fr, braham.othmen@virtuor.fr, hubert.zimmermann@ginkgo-networks.com

Abstract—The Internet has known an impressive success in the three last decades. 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, network virtualization 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 on 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 detail description of the proposed autonomic architecture and we focus on a real testbed 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

I. Introduction

Internet has known a great success in the last few decades. It is playing an important role in our society and it is used as the main way of communication and information exchange in the world. The number of Internet users grows exponentially every year. 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 [1][2][3].

So, innovative approaches are needed to overcome Internet "ossification" problem [2]. In this context, network virtualization provides a promising way to run multiple architectures simultaneously on a single infrastructure. It is a new concept that 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 [3][4]. Various architectures, experiments and services can be simultaneously supported by Virtual Networks (VN) [1][2][7].
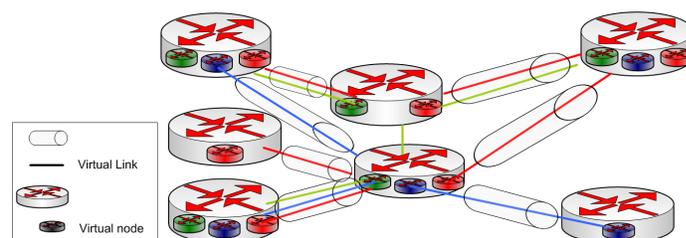

Fig. 1. Network virtualization model

As depicted in Figure 1, 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.

Network virtualization presents a diversified Internet architecture. It supports multiple coexisting virtual heterogeneous networks, sharing a common physical substrate. 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 [6].

In 2001, IBM proposes the "Autonomic Computing" paradigm [5] 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 [6] 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 main 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 summarizes the related work. In section 3, 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 describe the testbed setup and experimental results in section 4. Finally, section 5 concludes the paper and presents our ongoing work.

II. RELATED WORK

We find in the literature a number of systems and techniques that have been put forward to provision and manage virtual networks resources. For virtual network provision, authors propose greedy algorithms to efficiently assign VN to substrate resources. In [7], authors propose an embedding algorithm with admission control and online requests. This algorithm offers periodic re-optimization mechanisms like path splitting and path migration. Moreover, in [8], authors propose a method for mapping a virtual network in a cost-efficient way to ensure that instantiated virtual networks are able to handle any traffic pattern. Besides, [9] proposed an algorithm for virtual network assignments with dynamic reconfiguration.

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.

To manage virtual networks, many papers propose different primitives and mechanisms. [11] 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, [12] proposes techniques for dynamic allocation of processing resources (CPU) to virtual machines. Furthermore, in [13], 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 [14]. This approach provides
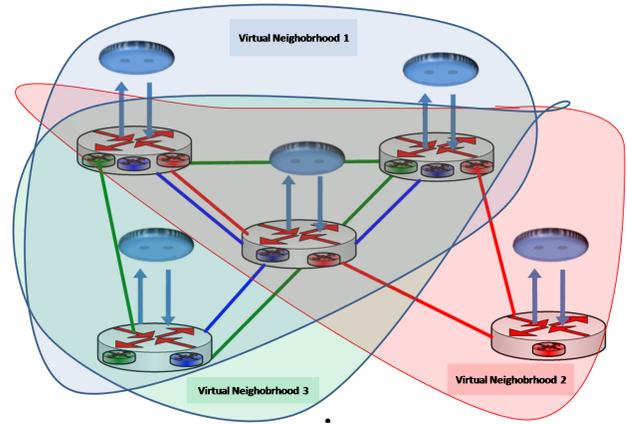

Fig. 2. Virtualized Network Infrastructure

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.

III. 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

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.

A. Network Infrastructure

As depicted in the Figure 2, 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".

B. Situated View

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 the Figure 2, an autonomic entity may have two types of neighbors:

- Physical neighbors which are neighbors that are physically connected to autonomic entities,
- Logical neighbors which are virtual neighbors. An autonomic entity maintains a neighborhood for each virtual network.

C. 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 un uncertain environment and possesses the autonomy to make decisions.

As shown in Figure 3, our architecture consists of the following components:

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.

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 [15] 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.

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 decision 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.

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 3 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.
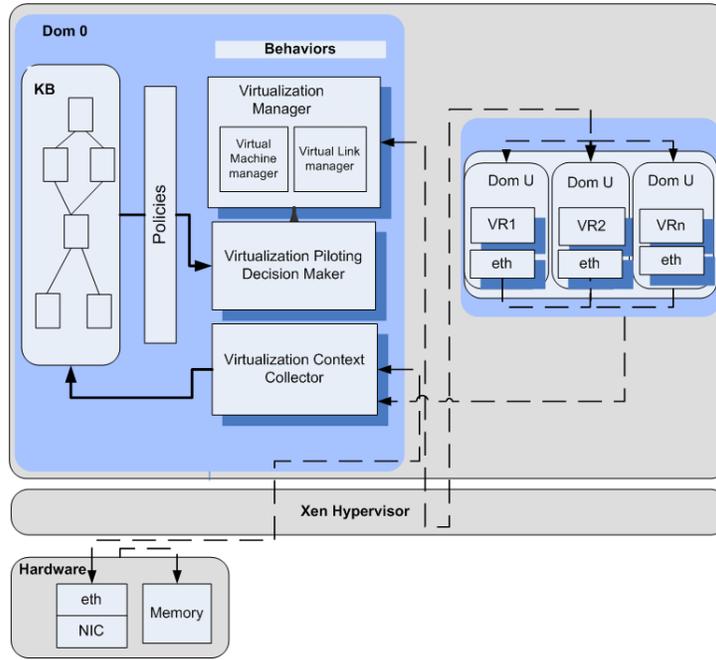
Fig. 3. Autonomic Piloting virtual Network Architecture

IV. 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 setup a real-world testbed instead of network simulations. In fact, due to implementation shortcuts 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.
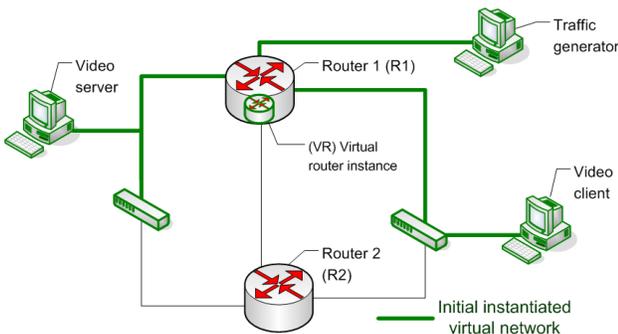
A. Experimental Setup

Fig. 4. Virtual Network Testbed

As depicted in Figure 4, the platform of our testbed consists of two routers with 4Go RAM, C2D-2.4 Ghz CPU and six 1Gbit/s network interfaces, embedding our proposed architecture and three end devices (a video server, a traffic generator and a video client) running GNU/Linux. A virtual network VN1 is instantiated between a video server and a video client. It passes through Router1(R1) and it is marked with green color in the Figure 4. 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.

B. Scenario and Results

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

Fig. 5. Quality of Video

We considered the following scenario. At t=30s, 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, the video displayed in client video 's screen is getting fuzzy (Fig. 5 (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:

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

where TimeIntervall=100ms

We note that based on extensive executions, we calibrated the error threshold to 100 packets/sec.

As soon as the rule becomes true, APVA 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 5 (b). Without adaptation, the VN1 would have probably crashed due to the lack of available resources.

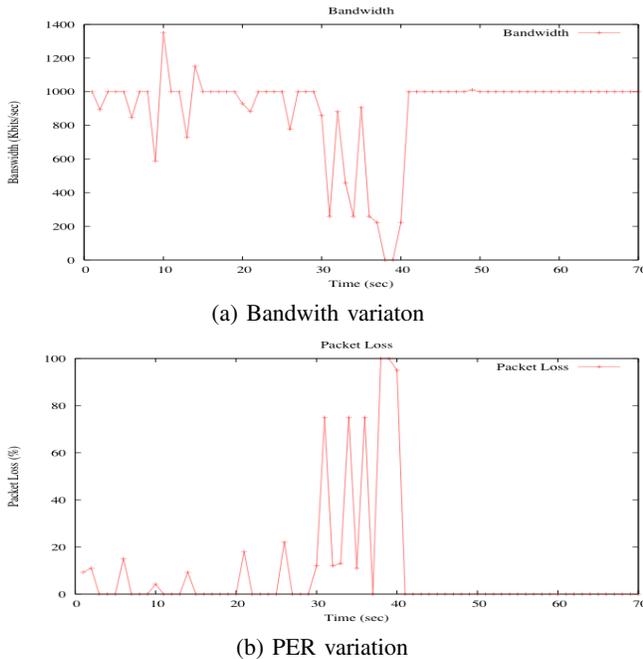(a) Bandwith variaton

(b) PER variation

Fig. 6. Network performance evaluation

Figure 6 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 6 is showing that APVA agent reacts to perturbation in less than 7s. In fact, APVA agent, decides to trigger migration only when PacketLossLevel exceeds 100 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.

As depicted in Figure 6, 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.

V. 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, analyze and then manage and optimize the use of network resources.

We have implemented the described autonomic system using Xen environment. First real experimentations presented in this paper are satisfying and prove the ability of our system to automatically reconfigure itself and improve its performances. More experiences are planned for future implementations to evaluate the performance of our proposal with different scenarios. We also plan to evaluate the performances of our system with large-scale experiments through simulations.

REFERENCES

[1] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, L. Mathy, *Network virtualization architecture: proposal and initial prototype*, VISA '09: Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, pp. 63-72, 2009.

[2] T. Anderson, L. Peterson, S. Shenker and J. Turner, *Overcoming the Internet Impasse Through virtualization*, IEEE Computer Magazine, vol 38,no.4, pp.34-41, 2005.

[3] J. Turner and D. Taylor, *Diversifying the internet*, in Proceeding if the IEEE Global Telecommunications Conference (GLOBCOM'05), vol. 2, 2005.

[4] N. Feamster, L. Gao, and J. Rexford, *How to lease the internet in your spare time*, SIGCOMM Computer Communication Review, vol. 37, no. 1, pp. 61-64, 2007.

[5] IBM Research, IBM and autonomic computing: an Architectural Blueprint for Autonomic Computing, IBM Publication, http://www.ibm.com/autonomic/pdfs/ACwpFinal.pdf, 2003

[6] P. Horn Autonomic computing: IBM's perspective on the state of information technology, also known as IBM's Autonomic Computing, 2001.

[7] M. Yu, Y. Yi, J. Rexford, and M. Chiang., *Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration*, SIGCOMM Comput. Commun. Rev., 38(2):17–29, 2008. Y. Zhu and M. Ammar.

[8] J. Lu, J. Turner., *Efficient Mapping of Virtual Networks onto a Shared Substrate*, Technical Report WUCSE-2006-35, Washington University, 2006.

[9] Y. Zhu and M. Ammar, *Algorithms for Assigning Substrate Network Resources to Virtual Network Components* in IEEE Infocom, 2006.

[10] N. M. M. K. Chowdhury, M. R. Rahman and R. Boutaba, *Virtual network embedding with coordinated node and link mapping*, IEEE INFOCOM, 2009.

[11] Y. Wang, E. Keller, B. Biskeborn, J. Merwe and J. Rexford, *Virtual Routers on the Move: Live Router Migration as a Network-Management Primitive*, IEEE SIGCOMM, 2008.

[12] D. Menasce and M. Bennani*Autonomic Virtualized Environments*, ICAS, 2006.

[13] P. Ruth, J. Dongyan Xu, R. Kennell and S. Goasguen, *Autonomic Live Adaptation of Virtual Computational Environments in a Multi-Domain Infrastructure*

[14] M. Kim Myung S. Kim, A. Tizghadam, A. Leon-Garcia and J. Won- Ki Hong, *Virtual Network based Autonomic Network Resource Control and Management System*, Proc. of IEEE Globecom, pp. 1075-1079, 2005.

[15] I. Fajjari, M. Ayari and G. Pujolle,*VN-SLA: A Virtual Network Specification Schema for Virtual Network Provisioning*, IEEE ICN, 2010