OpenFlow and On Demand networks

Mounira Msahli[1], Guy Pujolle[2], Ahmed Serhrouchni[1], Ahmad Fadlallah[1,3], Fouad Guenane[2],
[1]Télécom ParisTech, Paris, France
[2] LIP6 Laboratory, University of Paris 6, Paris, France
[3]Faculty of Computer Studies, Arab Open University, Beirut, Lebanon
{Mounira.Msahli, Ahmed.Serhrouchni, Ahmad.Fadlallah}@telecom-paristech.fr
{Guy.Pujolle, Fouad.Guenane}@lip6.fr

***Abstract*—The exponential growth of the Internet and the diversity of services delivered to the end users, such as Cloud computing, have catalyzed researches on new generation networks. Hence, several proposals have been made to change the network applications and Internet media; in particular network virtualization is in the center of research interest. It is widely considered given its ability of dynamic programming, energy saving and low cost. This promising paradigm boosts different researches in literature. Openflow, AKARI and 4ward projects are concrete examples. They follow common approaches of virtualization but with different characteristics. In this paper, we present an analysis and comparison between these projects. We mainly focus on Openflow solution to validate on demand virtual network architecture based on the Mininet simulator. This work is part of the On-Demand research project sponsored by European Regional Development Fund.**

***Index Terms*—Network Virtualization, On-Demand, Openflow, Mininet.**

I. Introduction

The concept of network virtualization is considered as a main way for Internet evolution [1]. It is defined by [2] as follows: "A Networking environment supports network virtualization if it allows coexistence of multiple virtual networks on the same physical substrate". This technology offers many advantages like reducing the cost of network deployment especially the Total Cost of Ownership (TCO), rapid deployment and adaptability. Therefore, a number of past and present researches on the issue have been done to develop new architectures and concepts such as the FEDERICA project [2,3,4], which combines network control mechanisms with virtualization techniques, and VINI project [5], which is considered as an instantiation of overlay networks. In addition, we find in the literature three most recent projects, named GENI [6] in USA, 4ward [7] in Europe and AKARI [8] in Japan.

GENI with its OpenFlow solution [9] adopts a clean slate approach [10]. This latter has proposed a completely redesigned architecture. It was considered the base element for new generation network researches. OpenFlow defines the virtualization as a slice of network resource on space and time.

Regarding the 4ward, it presents network virtualization as coexistence of several networks on the common physical infrastructure. Its main specificity is the presentation of virtual network business model with three actors: VNP (Virtual Network Provider), VNO (Virtual Network Operator) and InP (Infrastructure Provider).

With respect to AKARI project, it is based on the principle of Competition and Natural Selection. This later means that the best and optimal network virtualization solution is automatically selected from several competitive "clean slate" architecture propositions.
Among these projects mentioned above, only OpenFlow proposes its own simulator, called "Mininet"[11] in addition to many other network components such as Flowvisor (the network hypervisor) [12].

This paper analyzes and compares the mentioned projects (GENI, 4WARD and AKARI), with a main focus on
Openflow solution to validate on-demand virtual network architecture based on the Mininet simulator. It is organized as follows: Section 2 analyzes the three virtual network approaches proposed by Openflow, AKARI and 4ward. Section 3 shows a comparison between them based essentially on the flexibility, programmability and scalability. Section 4 presents the on demand network architecture. Section 5 describes the simulation of the on demand network architecture using open flow simulator Mininet combined with the openflow flowvisor. Finally, Section 6 concludes the paper and presents our forthcoming works.

II. Research approaches for network virtualization

In this section, we describe deeply the three projects already mentioned.

A. OpenFlow

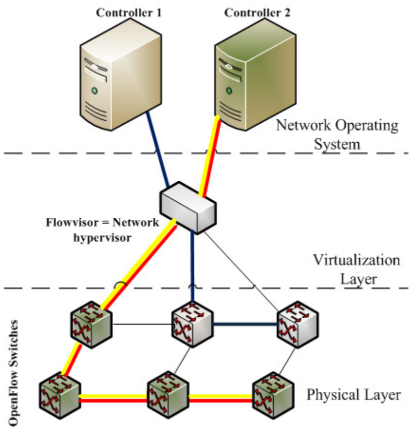

Fig. 1. OpenFlow network architecture

Initially, Openflow approach was the separation of two usual functions: control and forwarding (data path) often found in network devices such as router. Thus, there are two network components: the switch which guarantees the data forwarding and the controller that makes routing decisions. The communication between Controllers and switches uses the OpenFlow protocol [8] defining its own set of messages (packet-received, send-packet-out, etc.).

Openflow approach has moved to the network virtualization and consequently network hypervisor has been introduced, named flowvisor. As shown in Figure 1, the flowvisor plays the role of proxy between switches and controllers. Compared with the virtualization of operating systems, the flowvisor corresponds to the virtualization layer located between the hardware layer (OpenFlow-enabled switches) and the software layer (OpenFlow controllers).

The Flowvisor divides network into logical partitions called slices. One slice corresponds to the data flows running on switches' topology [13]. Slices work independently and separately. Each one controls only its own packet transmission and has a controller. Thus, the same hardware forwarding plane is shared by multiple logical or virtual networks.

Flowvisor takes into account quality of service and security constraints since it offers a number of isolation mechanisms like OpenFlow control isolation, bandwidth isolation, topology isolation, and flow entries isolation (it counts flow entries in each slice and ensures that it does not exceed a prefixed limit).

The idea of OpenFlow has led to concrete products: Open Flow Switch(for switches), NOX[15] (for controllers) and Flowvisor [12]. In addition, the OpenFlow Solution is validated using its own Mininet simulator. It is also worth-mentioning that several network equipment manufacturers (Cisco, HP, Juniper, etc.) have integrated the OpenFlow protocol in their products.

Finally, Open Networking Foundation (ONF) adopted the OpenFlow approach as its first standard for Software-Defined Networking (SDN) [16].

B. 4WARD

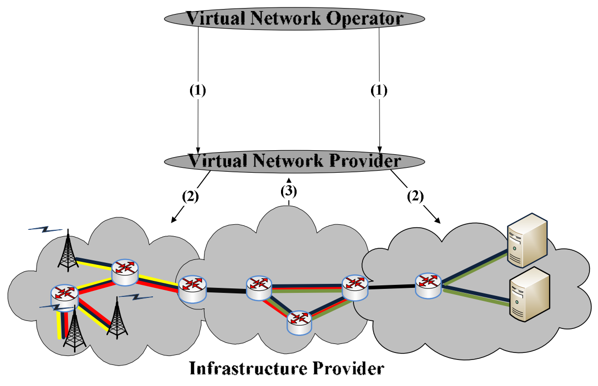

Fig. 2. Business model of virtual network

The 4ward project defines virtual network architecture related to three actors: Infrastructure providers (InP), Virtual Network Providers (VNP) and Virtual Network Operators (VNO) (see Figure 2).

- The Infrastructure Provider (InP) is responsible for the maintenance of physical resources like routers, network links, and wireless infrastructure. It enables virtualization on network equipment.
- The Virtual network provider (VNP) is in charge of virtual network construction using virtual resources or partial topologies provided by one or more InPs. It also ensures the virtual network management.
- The Virtual Network Operator (VNO) is responsible for Virtual Network configuration. It connects customers to virtual network services.

In order to precise the limits and the responsibilities of each actor, 4ward defines interactions between them:

- VNO can request a VNP to create a new virtual network and to modify or to withdraw an existing one. This is depicted with (1) in Figure 2.
- VNP can demand and negotiate network resources from InP. After resources allocation, it can request the installation of the virtual network and ask the migration of virtual nodes (between infrastructure providers). This is represented by the (2) in figure 2.
- Infrastructure provider can accept or decline the request of virtual network creation. This is shown by (3) in figure 2.

The highlight of 4ward virtualization approach is the ability to encompass wired and wireless networks. This project has dealt with numerous aspects of quality of service and mapping processes for assigning virtual networks to shared substrates networks.

The disadvantage of 4ward solution is the shortage of its implementation and validation. It roughly remains on the theoretical state.

C. AKARI

AKARI defines network virtualization as the technology that allows a shared physical core network to appear as multiple logical networks [17]. It enables users to construct, deploy, and evaluate multiple network architectures on a shared core network. For AKARI, There are three possible virtual network models, detailed in Figure 3:

- Isolated virtual network (Figure 3-a): its purpose is to enable the same physical resources to be shared by multiple independent and isolated logical networks.
- Transitive Virtual Network (Figure 3-b): its objective is to facilitate the migration from current architecture to the newly developed.
- Overlaid Virtual Network (Figure 3-c): It is defined as multiple and simultaneous architectures implemented using the same resources.

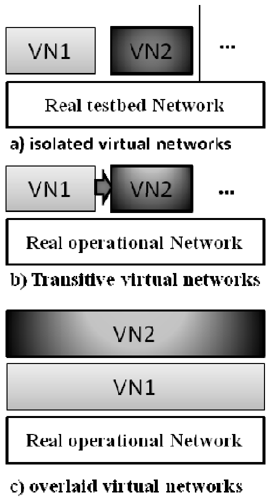

Fig. 3. AKARI architecture for network virtualization [8]

AKARI details four requirements to be considered in network virtualization architecture design:

- Determine in which layer virtualization will be performed.
- Resource management and the operational cost must be well measured because of the presence of distributed management plan shared by multiple cooperative virtual networks.
- Scalability mainly in isolated networks presented above.
- Virtualization should not only be limited to the network-layer but also studied in the physical layer. Since optical and wireless networks continue to be more familiar technologies in the future, it is suitable to investigate the problem regarding isolation and allocation of existing physical resources.

AKARI model is limited to a general description of the network virtualization architecture highlighting the requirements of its realization but without any precision on how to modulate or/and to implement this proposal.

III. COMPARISON OF DIFFERENT VIRTUAL NETWORK SCHEMES

This section presents a qualitative comparison between virtual network schemes described in section II.

Comparison criteria mentioned in Table-1 are enthused by other researches [1, 18, 19] and motivated by On-Demand project. As said before, the purpose of the project is to offer on demand network services using virtualization technologies and dynamic programming equipment.

4ward is the only project that defines the virtual network actors but it has unfortunately remained a theoretical model without any continuity. On the other hand, AKARI project presents rules that must be respected for designing efficient virtual network architecture. This talented architecture should verify certain requirements and takes into account the current problems of the Internet. Likewise, AKARI is still a theoretical proposal without implementation.

TABLE I. COMPARISON OF VIRTUAL NETWORK SCHEMES

Characteristics	4WARD	AKARI	OpenFlow
Scalability	Theoretical Model	Theoretical Model	Not finalized
Virtualization Layer	All layers	All layers	All layers
Virtualized infrastructure actors	(InP, VNP, VNO)	Not defined	Not defined
Standardization	VNRG-IRTF FGFN-ITU-T	ITU-T	OpenFlow ONF
Interoperability	Theoretical model	Theoretical model	Limited
Programmability	Programmable	Not defined	Programmable
Availability	Defined in theoretical model	Not defined	Not defined
Maturity	Medium	Medium	High
Virtualization Layer	All layers	Not defined	All layers
Products	No product	No product	OpenFlow components (Open vSwitch, NOX, etc…)

OpenFlow is the only solution that has implemented and imposed its model despite of some drawbacks concerning its scalability. This implementation has proved its feasibility, efficiency and maturity; that is why it has been approved by several most network equipment manufacturers like Cisco and Juniper.

Among all presented network virtualization models in table I, there is no solution that satisfies all required characteristics. In other words, there is no perfect model. Hence, it is very interesting to design a perfect model that answers all needs of network virtualization, but it should prove its feasibility. According to the latter sentence, OpenFlow seems to be the satisfactory solution for the future network virtualization. This justifies our choice for the openflow architecture to simulate the On demand network architecture.

IV. OUR ON DEMAND NETWORK ARCHITECTURE

The aims of On-Demand project are to design and to provision a network architecture that meets the following requirements:

- Virtualization: that takes into account the current network virtualization solutions in terms of maturity and efficiency,
- Programmability: dynamically reconfigurable architecture,
- Scalability: the ability to meet the growing demand of network resources.

In this section we propose on demand architecture presented in figure 4 that combines the advantages of the three schemes (4WARD, AKARI and OpenFlow) and responds to On-Demand project goals:

- It uses the implemented OpenFlow components that will be more detailed in the simulation section, hence based on Table I it is a programmable architecture,
- It takes into account the constraints proposed by AKARI. Indeed in our proposition, virtualization affects all layers; the scalability requirement will be validated in simulation section but the operational cost of this architecture is not treated in this article.
- It considers the definition of the virtual network actors provided by 4WARD:

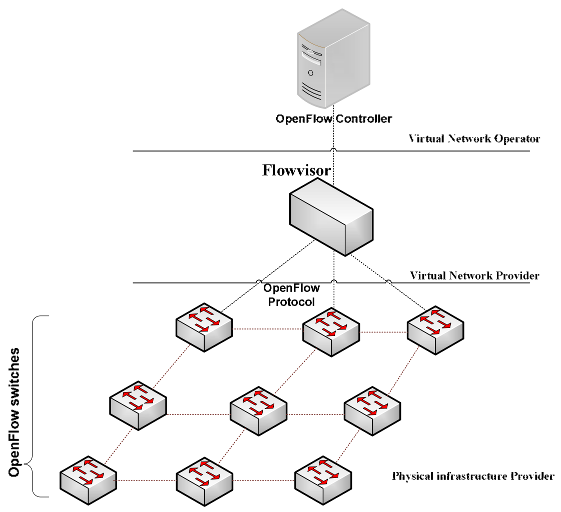

Fig. 4. On demand network architecture

The physical Infrastructure Provider (InP) is the responsible for the establishment, the maintenance and the configuration of OpenFlow enabled switches.

The Virtual Network Provider manages the flowvisor, sets slices and shares the logical management of switches with InP. It is also responsible for the network resources allocation to the Virtual Network Operator with the InP.

The controller is managed by the Virtual Network Operator (VNO), which connects the end-users to virtual network services.

This architecture satisfies On-Demand project requirements since it provides on demand network services using OpenFlow standard.

V. On demand architecture Simulation

This section presents the simulation of on demand network architecture (described in the previous section) using OpenFlow Simulator. It describes the simulation environment, and the simulation results.

A. Simulation Environment

As illustrated in figure 4, three Openflow components are needed: Controller, OpenFlow-enabled switch and flowvisor.

In OpenFlow, like in Software-Defined Network SDN, there is a separation between control and forwarding planes guaranteed by:

- Network operating systems, represented by Openflow Controllers like NOX[15], Beacon[20] and Maestro[21], control network using the functionalities involving access control, network virtualization and energy management.
- OpenFlow Switch consists of a secure channel to the controller and one or more tables, which ensure packet lookups and switching. It provides three features: forwarding packets' flow to a given port (or ports), encapsulating packets using secure channel [22] and sending it to the controller.
- The third component used in this simulation is the Flowvisor, responsible for the network virtualization by setting network resources slices and assigning their control to the controller. Controller's Slice policy fixes slice's rules.

Mininet is an Openflow prototyping environment, used to implement, test and validate an OpenFlow architecture in a large topology. The current Mininet version 1.0.0 can simulate only the Openflow switch and controllers because it does not contain the flowvisor software.

To overcome this problem, the best solution is to simulate a switches' network using Mininet and to connect it to flowvisor playing the role of remote controller. To realize this simulation, we install each component (Flowvisor and NOX controller) and the Mininet simulator in three different virtual machines and we connect between them.

B. Simulation Results

This simulation is the first step to validate the concept of on demand network architecture

TABLE II. Validation Set up time and end to end bandwidth

Switches Number	Set up time	End to End bandwidth
50	66.572 s	189 Mbits/s
200	273.003 s	50,4 Mbits/s
300	500.184 s	41.1 Mbits/s
400	834.043 s	28.6 Mbits/s

This simulation assumes that the flowvisor is already set and its configuration time is neglected. We create only one slice management which includes all network switches. Setup time is the time required to create Mininet Switches plus necessary time to make link between switches' network and flowvisor plus needed time to make link between flowvisor and NOX controller. These results give us an idea about On demand architecture deployment. As shown in this table, the time required for virtual network implementation is about few minutes. The main purpose of this simulation is to test scalability of our on demand architecture by varying number of switches deployed. The scalability criterion is maintained even with the slight degradation of the bandwidth.

VI. Conclusion

In this paper we have presented three virtual network proposals provided by AKARI, 4ward and Openflow projects. We have made a comparison between them based on several criteria such as maturity and programmability. OpenFlow is selected as the most appropriate solution for the On-demand project due to its deployability, maturity and programmability. Then we have presented On demand network architecture simulation with Openflow simulator "Mininet" and the software Flowvisor.

Simulation of this architecture is the first step in the On-demand project. It will actually be used to implement a secure private cloud. This will constitute our future works.

AKNOWLEDGEMENT

This work has been supported by the FEDER5 ONDEMAND project sponsored by European Regional Development Fund (http://www.network-ondemand.com).

References

[1] T.Anderson, L.Peterson, S.Shenker, J.Turner "Overcoming the Internet Impasse through Virtualization", Computer, vol. 38, no. 4, Apr. 2005, pp. 34–41.

[2] N.M. Mosharaf, Kabir Chowdhury, Raouf Boutaba A survey of network virtualization", Computer Networks: The International Journal of Computer and Telecommunications Networking , Avril 2010.

[3] P. Szegedi, S. Figuerola, M. Campanella, V. Maglaris, C. Cervell-Pastor, With evolution for revolution: the FEDERICA approach, IEEE Communications Magazine 47 (7) (2009) 34–39.

[4] P. Kauffman, M. Roesler, U. Monaco, A. Sevasti, S. Figuerola, A. Berna, J. Pons, D. Kagoleras, J.-M. Uze, P. Sjödin, M. Hidell, L.D. Cristina Cervelló-Pastor, R. Machado, Evaluation of current network control and management plane for multi-domain network infrastructure, FEDERICA Deliverable (2008) (Id: DJRA1.1).

[5] A. Bavier, N. Feamster, M. Huang, L. Peterson, J. Rexford, In VINI veritas: realistic and controlled network experimentation, in: Proceedings of the SIGCOMM'06, ACM, New York, NY, USA, 2006.

[6] L.Peterson, T. Anderson, D.Blumenthal,D.Casey, D.Clark,D.Estrin, J.Evans,D.Raychaudhuri, M.Reiter, J.Rexford, S.Shenker, John Wroclawski, GDD-06-08 : GENI: Global Environment for Network Innovations, August 2006.

[7] S.Baucke, C.Görg, M.Achemlal, T.Almeida, S.Baucke, R. Bless, M.Bourguiba, J.Mari, L.Caeiro, J.Carapinha, F.Cardoso L.Correia, M.Dianati, A.Feldmann, C.Görg, I.Grothues, I.Houidi, J.Jimenez, M.Kind, Y.Lemieux, W.Louati, L.Mathy O.Maennel, P.Papadimitriou, J.Sachs, S.Perez Sanchez, A.Serrador, I.Seskar, C.Werle, F.Wolff, A.Wundsam, Y.Zaki D. Zeghlache, L.Zhao, Virtualisation Approach: Concept, 4WARD Deliverable (2009) (Id: FP7-ICT-2007-1-216041-4WARD / D-3.1.1).

[8] H.Harai, M.Inoue, H. Otsuki, T.Kuri, K.Nakauchi, H.Furukawa, T. Miyazawa, S.Xu, V.P. Kafle, T.Umezawa, M.Ohnishi, K.Fujikawa, R.Li, M.Andre, S.Decugis, C.Peng, H.Yagi, M.Kamiya, M.Murata, F.Teraoka, H.Morikawa, A.Nakao, M.Ohta, H.Imaizumi, M. Hosokawa, T.Aoyama, New Generation Network Architecture AKARI Conceptual Design,AKARI Deliverable (2009).

[9] N.McKeown, T.Anderson, H.Balakrishnan, G.Parulkar, L.Peterson, J.Rexford, S.Shenker, J.Turner, OpenFlow: Enabling Innovation in Campus Networks, ACM SIGCOMM Computer Communication Review, Volume 38, Number 2, April 2008

[10] A.Feldmann, Internet clean-slate design: what and why, ACM SIGCOMM Computer Communication Review, Volume 37 Issue 3, July 2007.

[11] B.Lantz, B.Heller, N.McKeown, A Network in a Laptop: Rapid Prototyping for Software-Defined Networks, permission, Hotnets '10, ACM, October 2010, USA.

[12] R. Sherwood, G.Gibby, K.Yapy, G.Appenzellery, M.Casado, N.McKeowny, G.Parulkary, FlowVisor: A Network Virtualization Layer, October 2009,USA.

[13] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, S. Shenker, "Extending Networking into the Virtualization Layer," HotNets-VIII, Oct. 22-23, 2009.

[14] J.Naous, D.Erickson, G. Adam Covington, G.Appenzeller, N.McKeown, Implementing an OpenFlow switch on the NetFPGA platform, in: ANCS '08 Proceedings of the 4th ACM/IEEE,ACM,USA,2008 .

[15] N.Gude, T.Koponen, J.Pettit, B.Pfaff, M.Casado, N.McKeown, S.Shenker, NOX: Towards an Operating System for Networks, ACM SIGCOMM Computer Communication Review, Vol 38, Number 3, July 2008.

[16] J.Rexford, Programming languages for programmable networks, POPL '12 Volume 47 Issue 1, page 215-216 ACM, USA, January 2012.

[17] T.Aoyama, A New Generation Network: Beyond the Internet and NGN, IEEE Communications Magazine, May 2009.

[18] B. Belter, M. Campanella, F. Farina, J. Garcia-Espin, J. Jofre, P. Kaufman, R.Krzywania, L.Lechert, F. Loui, R. Nejabati, V. Reijs, C. Tziouvaras, T. Vlachogiannis, D. Wilson, Virtualisation Services and Framework – Study, GÉANT Delivrable (2012). (Id: GN3-12-123).

[19] B.Melander, V.Souza, V.Fusenig, A.Sharma, M.Meulle, D.Audsin, P.Murray, S.Sae Lor, L.Vaquero, T.Begin, P. Gon_calves, G.Koslovski, S.Roy, W.Louati, M.Mechtri, H.Medhioub, D.Zeghlach, M.Hidell, R.Stadler, P.Sjodin, D.Turull, F.Wuhib, P.Vicat-Blanc, D.Dudkowski, J.Carapinha, M.Melo, J.Soares, R.Monteiro, D.Gillblad, R.Steinert, B.Bjurling, B.Levin, A.Miron, P.Aranda, I.Menem, Cloud Network Architecture Description, SAIL Deliverable (2011), (FP7-ICT-2009-5-257448-SAIL/D-D.1).

[20] A.Voellmy, J.Wang, Scalable Software Defined Network Controllers, SIGCOMM'12,ACM, Finland August 2012

[21] Z.Cai, A.L.Cox,T.S.Eugene, Maestro: A System for Scalable OpenFlow Control, 21st Large Installation System Administration Conference (LISA '07), USA, 2007.

[22] B.Pfa, B.Lantz, B.Heller, C.Barker, D.Cohn, D.Talayco, D.Erickson, E.Crabbe, G.Gibb, G.Appenzeller, J.Tourrilhes, J.Pettit, K. Yap, L.Poutievski,M.Casado, M.Takahashi, M.Kobayashi, N.McKeown, P.Balland, R.Ramanathan,R.Price, R.Sherwood, S.Das, T.Yabe, Y. Yiakoumis, Z.Lajos Kis. OpenFlow Switch Specification (Version 1.1.0 Implemented), (2011).