A SDN-based Network Virtualization Architecture with Autonomic Management

Qinglei Qi, Wendong Wang, Xiangyang Gong, Xirong Que
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing 100876, China
{qql,wdwang,xygong,rongqx}@bupt.edu.cn

***Abstract*—The ossified architecture and widespread application of the internet have restricted its development. For overcoming the existing obstacles of network innovation and management, network virtualization and autonomic management have been proposed. However, these two attractive technologies are subjected to the headaches of updating various devices. Software-Defined Networking (SDN) separates the control plane from the data plane. Therefore, some network control and management functions could be implemented through softwares. This provides a chance for the network virtualization and autonomic management. In this paper, a SDN-based network Virtualization architecture with Autonomic management (SDNVA) is proposed. The SDNVA architecture enables heterogeneous virtual networks to coexist on the same physical network, and supports hierarchical autonomic management. Moreover, we evaluate SDNVA on our testbed and find that the SDNVA architecture can perceive topology changes and isolate multiple virtual networks.**

***Keywords*—Network virtualization, Autonomic network management, Software-Defined Networking**

I. Introduction

The flattening architecture has been booming the internet tens of years ago. However, this architecture faces lots of challenges. Among of these challenges, the rigidity of the architecture restricts the incremented change of the internet, and the spread scope of the internet complicates network management reversely. To cover the ossification of the internet, the Network Virtualization (NV) technology is proposed. From the point of view of research, NV satisfies the requirements for large-scale and realistic experiment environments while the production services are not effected. For example, the PlanetLab [1], which currently comprises 1204 nodes at 593 sites, has assisted more than one thousand researchers from top academic institutions and industrial research labs. Moreover, by exploiting NV, various service providers can obtain their dedicated network from the same physical infrastructure. Therefore, the network resource of the Internet Service Provider (ISP) is utilized effectively and the small Content Provider (CP) doesn't worry the problem of lacking money to invest their private network anymore.

On the other hand, to deal with the complex network management, many research institutes and enterprises focus on the Autonomic Network Management (ANM). The ANM system can adaptively adjust the allocation of network resources for various flows and users without or with a little help of network manager. ANM is not a novel technology for alleviating management burdens of the operator. IBM proposes the MAPE-K [2] model first for the management of computer systems. Referencing to the MAPE-K model, the fully distributed [3] and the mix of distributed and centralized [4] ANM architecture are proposed. In these architectures, almost all of autonomic elements contain also the monitoring, analyzing, planning and executing components, which have the similar functions with them in the MAPE-K model. However, ANM is more difficult than the management of computer systems. The reason is that heterogenous management domains may conflict each other when they are connected [5].

Software-Defined networking (SDN) is another attractive and promising solution for the network control and management. SDN separates the control plane and the data plane and abstracts the hardware details of the routers and switches. These characteristics enable administrator to manage a network as a computer. So the approach of the computer virtualization can be considered to be used for the NV in SDN – i.e., a software is developed on the network operate system for providing a integrated virtual network environment for each tenant. Then each tenant designs private management applications used to the virtual network environment. The centralized architecture of SDN also is a chance for ANM. The realization of ANM needs updating lots of node in traditional network, whereas it just need the development of corresponding applications in SDN.

In this paper, we propose a SDN-based network Virtualization architecture with Autonomic management (SDNVA), in which a virtualization component is designed as a "base" application of NOX for slicing the OpenFlow network, and the autonomic management applications for the physical and virtual networks are design based on the NorthBound (NB) interface of NOX.

978-1-4799-7470-2/14/$31.00 ©2014 IEEE

These autonomic management applications follow the hierarchical control loop structure.

This paper is organized as follows. Section II presents related work on NV and ANM. Section III details the proposed SDNVA architecture. Section IV describes the testbed and the experimental work. Finally, Section V concludes the paper.

II. Related work

Various openflow-based network virtualization architectures or platforms have been proposed. Advisor [6] and FlowVisor [7] is a transparent agent laying between OpenFlow switches and controller. They divide logically the network resource into multiple slices, one slice per virtual network. However, these middleware-based virtualization approaches lack the flexibility of adjusting the virtualization strategy. In SDNVA, the virtualization component can be used by high-level application to slice network agilely. The fact that NOX can handle multiple flow-initiations concurrently and OpenFlow [8] supports multiple flow tables and MPLS provides a basement of the NV. Nevertheless, the original NOX doesn't separates the topology and traffic for the different services. CP doesn't have the opportunity to make private strategies for network control. SPARC [9] provides abstracted network view to the network application plane using a basic set of filter function on the network operate system. The abstract network can prevents the leakage of network information while guaranteeing the requirements of the management application. This architecture is similar with our work. The difference is that SPARC doesn't support the user-defined transport protocol and PindSwitch [10], a protocol-independent flow processing platform proposed in our previous work, can be supported in SDNVA.

As mentioned in the introduction, most of the ANM architectures obey the control loop structure exploited in MAPE-K model. Moreover, GANA adopts the hierarchical control loop structure on the decision plane. The decision elements are categorized into different decision level based on their decision domain. This hierarchical control loop structure is referenced in SDNVA. The ANM mechanism in SDNVA is divided into two layers: autonomic virtual and physical network management.

III. SDNVA Architecture

The SDNVA architecture is illustrated as in Fig. 1. The physical network controller (C_p) hosts multiple virtual network controllers (C_{vx}). The virtual network switch (S_{vx}) is embedded in the Physical network Switch (S_p). Each controller controls all the switches in its domain and doesn't access other domain. For example, C_p controls all the physical network switches, and C_{v1} controls all the virtual switches denoted by S_{v1}. In this paper, the virtual networks encompassing the controller and the switches are isolated by the virtualization component. In advance, we add the transport protocol management component for satisfying the NV requirements of diversified services. The Autonomic Management Applications can supervise the corresponding domain autonomically through the NorthBound (NB) interface of the controller and AMA_p can guide each AMA_{vx}. The virtualization component, the transport protocol management component and the autonomic management applications are presented in detail in following subsections.

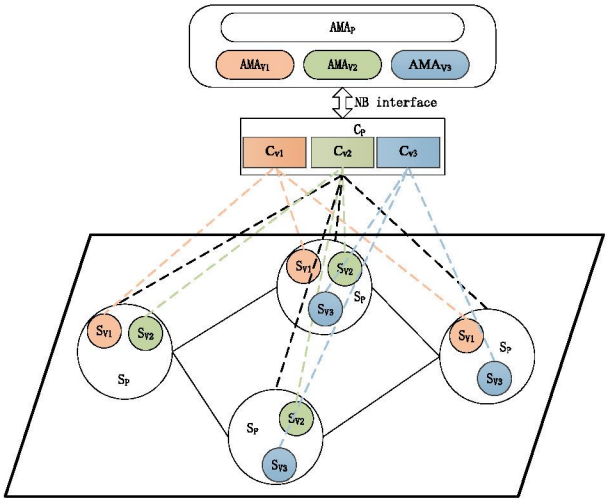

Fig. 1. SDNVA Architecture.

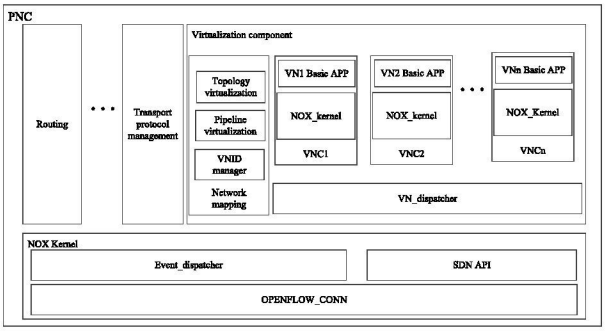

Fig. 2. Extended NOX controller.

A. Virtualization component

As shown in Fig.2, the virtualization component is a supplementary "base" application of the NOX which enables physical network to be transparency to the virtual network controller. It includes VN_dispatch and network mapping module. The main functions of the virtualization component are the initialization of virtual network and to handle the event associated with NV.

1) Initialization of virtual network

The virtualization component listens on the designated port (the 6644 port is used in this paper.) for the request of new virtual network. Once this component receives a request for initialization of the virtual network controller, it initializes a thread as the virtual network

controller and launches the network mapping model. The network mapping model generates a Virtual Network IDentification (VNID) and a network view for the virtual network based on the supervision of high layer application. The network view is in fact the composition of network topology, pipelines of OpenFlow switches in this network topology and other network resources (e.g. link bandwidth). As shown in Fig. 3, the pipeline virtualization is accomplished through configuring the first flow table in the switch.

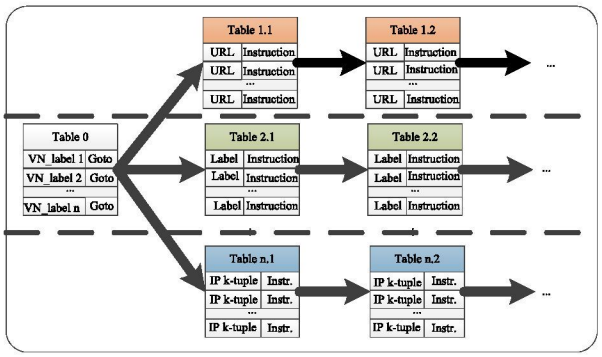

Fig. 3. pipeline virtualization.

2) Event handling mechanism for virtual network

According to the event mechanism in NOX, the virtualization component also receives event from Event_Dispatcher in the NOX kernel. After when virtualization component receives the event from the Event_dispatcher, the network mapping module extracts the information corresponding to the virtual network from the event. The translation result is delivered to the VN_dispatcher, and then the VN_dispatcher forwards this event to the virtual network controller. The response process to the event is reverse. The virtual network controller delivers the information to VN_dispatcher, and this information is delivered to the network mapping module. The network mapping module transfers the information into the information corresponding to the physical network. The result is sent to the Event_dispatcher. These event generated by OpenFlow messages of virtual network and high layer applications.

B. Transport protocol management component

In the process of initialization of and event handling for virtual network, VNID is the only identification of the virtual network. VNID can be tagged based on the 12-tuple supported by OpenFlow in NOX. However, 12-tuple can not denote new protocol (e.g. URL used in ICN). For supporting new protocol stack that is customized by network tenant, we supplement another "base" application of the NOX, the transport protocol management component (shown in Fig.2) which take charge of the register and configuration the transport protocol. In this component, the protocol stack is described by NetPDL language, and the description of each protocol layer contains three parts: the basic information, the definition of all the fields and the pointer to upper-layer protocol.

C. Hierarchical autonomic management applications

The management layer of SDNVA follows a hierarchical model, which includes autonomic management applications for the physical and virtual network from the top down. The autonomic management application

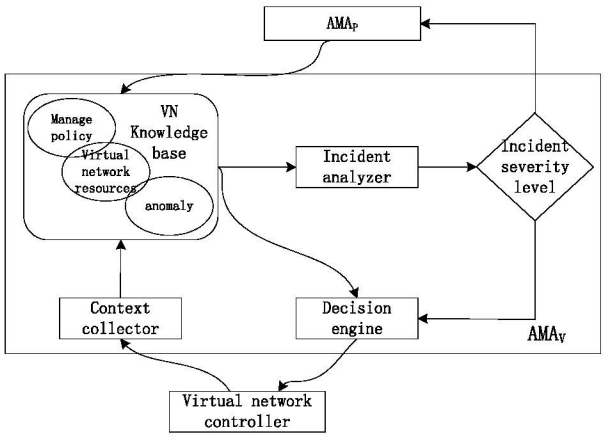

Fig. 4. Autonomic Management Mechanism for Virtual Network.

for the virtual network (AMA_v) (Fig. 4) has three main functions. The first function is periodically to analyze the virtual network context until the failure or optimization opportunities appear. The second function is to estimate whether the incident can be solved by itself. If AMA_v has not the ability to solve the incident, this incident has to be reported to the autonomic application for physical network management (i.e., AMA_p). For example, there are too many flows in the virtual network so that the virtual network resource cannot guarantee the QoS requirement of the service. AMA_v has to request more network resource from AMA_p. The third function is to decide the reasonable configuration policy for the virtual network controller.

The autonomic management application for the physical network (AMA_p) (Fig. 5) has similar functions with AMA_v. The different is that the action object of AMA_p is the physical network controller and AMA_p informs the network manager when it finds the incident beyond its ability. In addition, AMA_p must respond to the request from AMA_v or notify the virtualization component of the change of the physical network sate actively. For example, when AMA_p check the link or switch failure firstly, it informs this incident to the virtualization component. Then the virtualization component adjusts the topology mapping. For highlighting the relationship between AMA_p and AMA_v, the physical

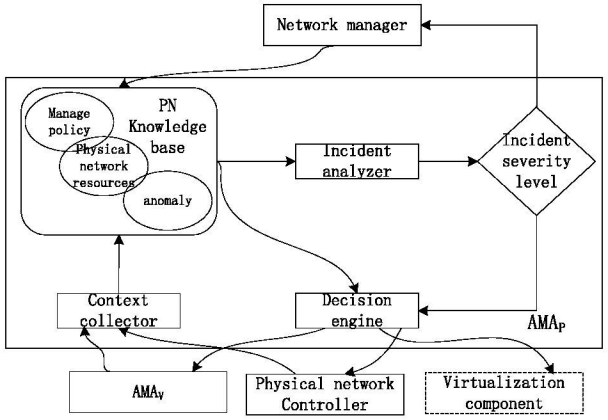

Fig. 5. Autonomic Management Mechanism for Physical Network.

network controller in this paragraph is exclusive of the virtualization component.

IV. Evaluation

The topology of the test network is illustrated by Fig.6. It consists of seven OpenFlow enabled switches, one controller, three PC clients and one server. The server provids the SIP and Web service. Two of the three clients are SIP clients, and the other one is Web clients. Three experiments for verifying the ability of SDNVA are performed. In the first experiment, The topology changes of the virtual network are observed when some switches are closed. In the second and third experiments, It is shown that the flow tables are configured correctly when the web and SIP services are deployed on the different virtual networks respectively. For showing the results briefly, Only the typical flow tables are presented and the columns whose values are wildcards are omitted in these flow tables.

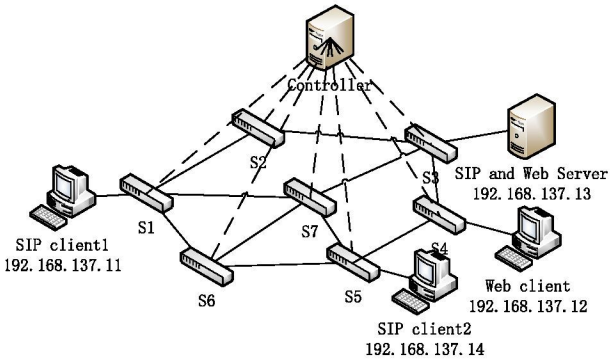

Fig. 6. SDNVA testbed.

A. Topology discovery

For viewing the network topology intuitively, we develop the topology display service based on the topology component in the NOX. The resource mappings for two virtual networks are configured initially. One virtual network consists S1, S2, S3, S4 and S7, while the other virtual network consists S1, S3, S5 and S7. After the Controller launched, the topologies of the physical network, virtual network 1 and virtual network 2 are presented as Fig7 (a), (b) and (c) respectively. When S3 and S5 are turned off, the topologies of the aforementioned three networks are shown as in Fig 7 (d), (e) and (f) respectively. In these figures, the red number denotes switch ID, the black number denotes the interface ID of the switch, and the light color nodes and the dotted lines denote the switches and links that exist in the physical network, but doesn't exist in the virtual network, respectively.

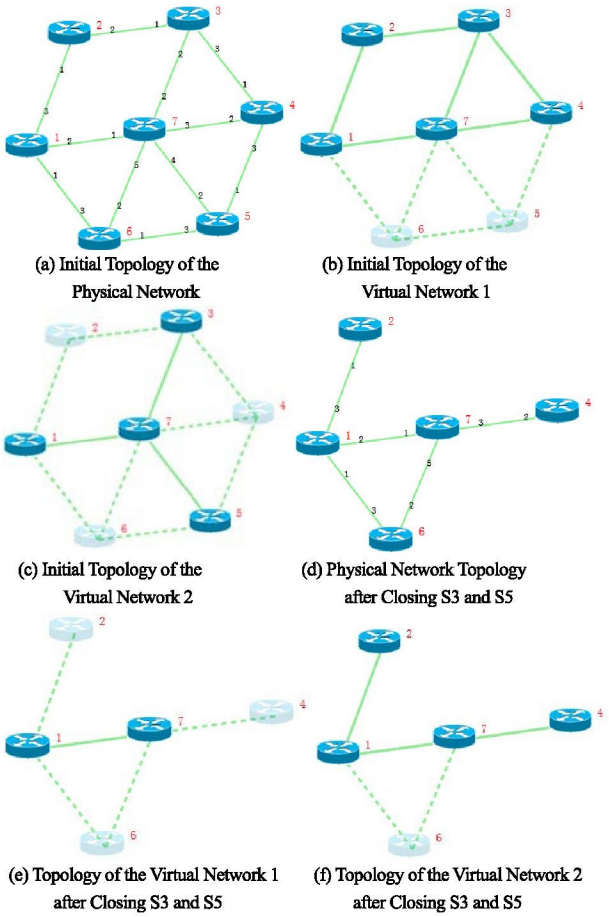

Fig. 7. Topology-Aware Mapping between the Physical and Virtual Networks.

B. Virtual network for the web service

The virtual network 1 (composed by S1, S2, S3, S4 and S7) carries the Web service. We started a web request from web client to Web Server and succeeded to obtain the test page. In the S4, the flow table 0 is configured two entries (Table III). One entry rules that the packets coming from virtual network 1 are forwarded to the flow table 10 (Table IV) while the other entry rules that the other packets are forwarded to the flow table 1 (Table V). The entries in the Flow table 1 rule that the IP packet from the web client is added a VLAN header

TABLE I
THE FLOW TABLE 20 OF S1.

priority	eth type	MPLS label	ip src	ip dst	action
32678	0x800	-	192.168.137.11	192.168.137.13	push-MPLS ethertype 0x8847, set-field mpls-lable 119, output port 2
32678	0x800	-	192.168.137.11	192.168.137.14	push-MPLS ethertype 0x8847, set-field mpls-lable 25, output port 2
32678	0x8847	131	-	-	pop-vlan, pop-mpls ethertype 0x800, output port 5
32678	0x8847	203	-	-	pop-vlan, pop-mpls ethertype 0x800, output port 5

TABLE II
THE FLOW TABLE 0 OF S7.

priority	switch port	eth type	MPLS label	action
32678	1	0x8847	119	set-field mpls-lable 182, output port 2
32678	2	0x8847	73	set-field mpls-lable 131, output port 1
32678	1	0x8847	25	set-field mpls-lable 77, output port 4
32678	4	0x8847	138	set-field mpls-lable 131, output port 1

TABLE III
THE FLOW TABLE 0 OF S4 .

priority	VLAN ID	action
32678	1	goto-table 10
0	-	goto-table 1

and forwarded to the flow table 0 or the start of the pipeline. The entry in the flow table 10 rules that the IP packet destined for the web client is taken off the VLAN header and output from the port 4, and the the IP packet destined for the web server is output from the port 1.

TABLE IV
THE FLOW TABLE 1 OF S4.

priority	eth type	ip src	action
32678	0x800	192.168.137.12	push-vlan ethertype 0x8100, set-field vlan-id 2, output port TABLE

TABLE V
THE FLOW TABLE 10 OF S4.

priority	eth type	ip dst	action
32678	0x800	192.168.137.12	pop VLAN, output port 4
32678	0x800	192.168.137.13	output port 1

C. Virtual network for the SIP service

The virtual network 2 (composed by S1, S3, S5 and S7) carries the Web service. We launch a SIP call from the SIP client 1 to the SIP client 2 and succeed to establish the connection. The contents of the flow table 0 and 1 in S1 are similar with the contents of the flow table 0 and 1 in S4 mentioned above. In S1, the entries in the flow talbe 0 rules that the packets coming from virtual network 1 is forwarded to the flow table 20 while the other packets is forwarded to the flow table 1. The entries in the flow table 1 rules that the IP packet from the SIP client 1 is added a VLAN header and forwarded to the flow table 0 or the start of the pipeline. The entry in the flow table 20 (Table I) rules that the packets being forwarded from the SIP client 1 to the SIP server and SIP client 2 are added MPLS header and forwarded to S7, and the packets being forwarded to the SIP client 1 are taken off the VLAN and mpls header. In S7, the flow table 0 (Table II) rules that all the packets are operated based on the mpls lable.

From the results presented above, we can see that our SDN architecture can realize the network virtualization and perceive the changes of the network status.

V. CONCLUSION

In this paper, we have presented the design of SD-NVN. The major objects of SDNVN are to isolate the virtual network resources using a virtualization component like sand box and to provide a hierarchical autonomic management model for the virtual and physical network. We have demonstrated the availability of SD-NVN in our testbed. SDNVA enables the virtual network to respond to the changes of the physical network. The forecasting for the anomaly or risk will decrease the loss of time and money, which makes great sense for the users, service and content providers. This is the next step of our work. We will develop advanced autonomic application, which has the ability of learning and forecasting for new incidents.

ACKNOWLEDGMENT

This work was supported in part by the National High Technology Research and Development Program (863 Program) of China under Grant No.2013AA013301, No.2013AA013303, and No. 2011AA01A101.

REFERENCES

[1] Plantlab, https://www.planet-lab.org/node/1.
[2] Kephart J O, Chess D M. The vision of autonomic computing[J]. Computer, 2003, 36(1): 41-50.
[3] Bouabene G, Jelger C, Tschudin C, et al. The autonomic network architecture (ANA)[J]. Selected Areas in Communications, IEEE Journal on, 2010, 28(1): 4-14.
[4] Chaparadza R, Papavassiliou S, Soulhi S, et al. The Self-Managing Future Internet powered by the current IPv6 and extensions to IPv6 towards "IPv6++"-A viable roadmap Scenario for the Internet Evolution Path[C]//GLOBECOM Workshops (GC Wkshps), 2010 IEEE. IEEE, 2010: 551-556.
[5] Movahedi, Z., et al. (2012). "A survey of autonomic network architectures and evaluation criteria." Communications Surveys & Tutorials, IEEE 14(2): 464-490.
[6] Salvadori, Elio, et al. "Demonstrating generalized virtual topologies in an openflow network." ACM SIGCOMM Computer Communication Review. Vol. 41. No. 4. ACM, 2011.
[7] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. "Flowvisor: A network virtualization layer." Technical Report Openflow-tr-2009-1, Stanford University, Tech. Rep., 2009.
[8] B. Pfaff,et al., OpenFlow Specification, Version 1.2.0, https://www.opennetworking.org/images/stories/downloads/specification/OpenFlow-spec-v1.2.pdf
[9] SPARC deliverable 2.2: revised definition of use cases and carrier requirements, June 2012, http://www.fp7-sparc.eu/assets/deliverables/SPARC_D2.2_Revised_definition_of_use_cases_and_carrier_requirements.pdf.
[10] Zhou T, Xiangyang G, Hu Y, et al. PindSwitch: A SDN-based protocol-independent autonomic flow processing platform[C]//Globecom Workshops (GC Wkshps), 2013 IEEE. IEEE, 2013: 842-847.