

Implementation of an OpenFlow Network Virtualization for Multi-Controller Environment

Seokhong Min, Seungju Kim, Jaeyong Lee, Byungchul Kim*, Wontaek Hong, Jonguk Kong**

* Department of Information & Communications Engineering, Chungnam National University, Daejeon, Korea

** Korea Institute of Science Technology and Information, 245 Daehangno, Yuseong-gu, Daejeon, 305-806, Korea

{minsh, ohhara, jyl, byckim}@cnu.ac.kr, {wthong, kju}@kisti.re.kr

Abstract— To share Future Internet OpenFlow testbed with multi-users using multi-controllers without each other's intervention, network virtualization based on FlowVisor should be implemented. For network resource virtualization, bandwidth isolation at allocated switches on a given slice and admission control are required. In this paper, we implemented admission control and minimum bandwidth guaranteeing scheme at the FlowVisor and tested the transmission performance of QoS-video streams on this testbed. Using our enhanced FlowVisor, we show that each user's bandwidth requirement can be guaranteed irrespective of other users' flow traffic.

Keywords— OpenFlow, Future Internet, Network Virtualization

I. INTRODUCTION

Current Internet based on best-effort delivery concept has been confronted with many problems on accepting new emerging future services. Also, it has many shortcomings in scalability, security, virtualization, quality of service, and so on. To overcome these problems, new research activities on the Future Internet are being actively performed in many countries such as FIND[1], FIA[2] and NwGN[3]. The research directions can be classified into two major categories: One is to develop new architecture and protocols, and the other is to deploy Future Internet testbed for running newly designed protocols. Most well-known testbed researches are GENI (Global Environment for Network Innovation)[4] in US, FP7 FIRE (Future Internet Research and Experimentation)[5] in Europe and NwGN (New Generation Network)[3] in Japan.

Among them, one of the most widely deployed testbed is the OpenFlow network suggested by Stanford University[6]. OpenFlow protocol provides a standard way to control flow tables in commercial switches and routers. So researchers can control their own flows by choosing the routes and processing rules to apply their new network protocols. On this testbed, it's very important to convince each researcher to believe that he only uses assigned resources without other users' intervention. To do this, the resource on the OpenFlow testbed should be shared in a controlled and isolated way by using network virtualization scheme.

The network virtualization concept is defined as the process of combining hardware and software network resources and network functionality into a single software-based administrative entity, a virtual entity, according to the Wikipedia[7].

For virtualization of OpenFlow network and flow isolation, network virtualization layer called FlowVisor[8] should be deployed between OpenFlow controllers and switching devices. FlowVisor hosts multiple OpenFlow controllers and acts as a Proxy controller to the OpenFlow devices. So, it intercepts messages between OpenFlow switches and controllers. One controller is allocated to each slice, i.e. sets of flows running on a topology of switches, and controls its own slice. Each slice can be isolated from another. Next, network resources such as bandwidth, topology, device CPU and flow table should be isolated. Adding admission control function to FlowVisor and scheduling function to switches, we can allocate required bandwidth to a specified slice.

In Korea, OpenFlow-based Future Internet testbed has been deployed and integrated by linking OpenFlow-enabled resources managed by several research institutions and universities. The interconnection of each resource domain was implemented using VLAN stitching. Now the enhanced FlowVisor needs to be deployed to provide network virtualization by implementing admission control and scheduling functions.

In this paper, we developed an enhanced FlowVisor for admission control and scheduling the bandwidth resource. These functions can help dynamic resource control based on user's requirement and network conditions. Also, the quality of video steaming was tested under QoS guaranteeing scenario and its performance is compared to non-QoS supporting video stream.

The paper is organized as follows: Section 2 introduces OpenFlow and FlowVisor function. Section 3 explains the detailed implementation of our proposed admission and scheduling scheme. Section 4 shows the performance of video streaming test. Section 5 gives conclusions and future work.

II. FLOWVISOR AND OPENFLOW PROTOCOL

OpenFlow protocol enables the controller to access the data-path of switch by controlling flow table. The structure of OpenFlow switch is shown in Fig. 1. OpenFlow switch consists of flow table, secure channel for communication with external controller and OpenFlow protocol. The data-path of all the packets on the OpenFlow network is controlled by OpenFlow controller. A flow is defined by the combination of following 10-tuples; switch port, source and destination MAC

address, ethernet type, VLAN ID, source and destination IP address, IP protocol field, and source and destination TCP port number.

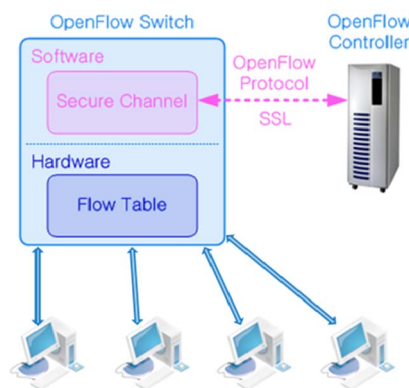


Figure 1. OpenFlow Switch

OpenFlow controller is a kind of network operating system for controlling OpenFlow network. It provides centralized programmable interface. NOX[9] is an open source type OpenFlow controller and recent information can be obtained from <http://www.noxrepo.org>. NOX has lots of useful applications for user-defined network control based on the network view. OpenFlow switch communicates with controller whenever new switch is connected to the network or a new flow enters into the network. The detailed NOX installation and configuration commands are as follows.

```
git clone git://noxrepo.org/noxcore
cd noxcore/
./boot.sh
mkdir build/
cd build/
./configure --with-python=yes
make
make check
```

When a number of experimenters want to control network resources independently using their own controller, FlowVisor should be deployed for network virtualization. The network structure using FlowVisor is shown in Fig. 2, where it acts as transparent Proxy between network devices and OpenFlow controllers (or other FlowVisors).

Thus, FlowVisor is the only equipment that can get all the information of OpenFlow devices and slice running on the testbed. To guarantee user's requirement and bandwidth isolation, admission control should be implemented on top of basic FlowVisor functions using these information. Also, scheduling function at OpenFlow devices and required parameters should be configured from the FlowVisor. The basic FlowVisor functions should be extended by implementing admission and scheduling modules, and we call it 'enhanced FlowVisor' from now on. In next section, we'll explain its structure in detail.

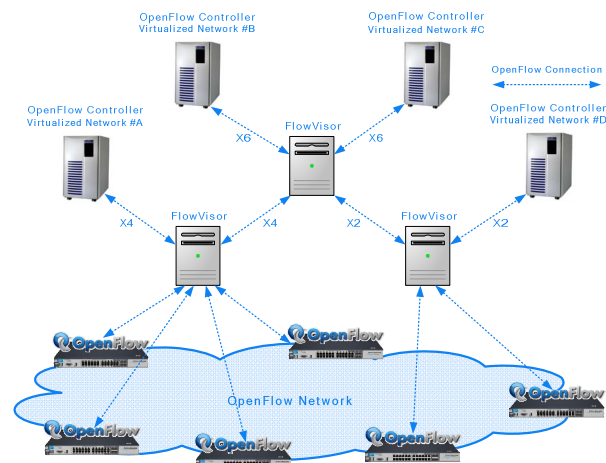


Figure 2. FlowVisor

III. ENHANCED FLOWVISOR

Enhanced FlowVisor extends the basic FlowVisor functions by implementing admission control, VLAN PCP setting module, and switch scheduler activation module. We used HP Procurve OpenFlow switches and OpenFlow protocol version 1.1. OpenFlow HP switches provide guaranteed minimum bandwidth (GMB), so we can allocate required bandwidth to a specified flow by assigning a proper VLAN PCP field. Assume that VLAN ID of a flow on the assigned slice is set to m and required bandwidth is set to 100 Mbps, then FlowVisor sends an OpenFlow message to modify PCP field of VLAN ID- m traffic to some value (e.g., k) and configure OpenFlow switches to guarantee minimum bandwidth 100 Mbps for the PCP- k flow traffic.

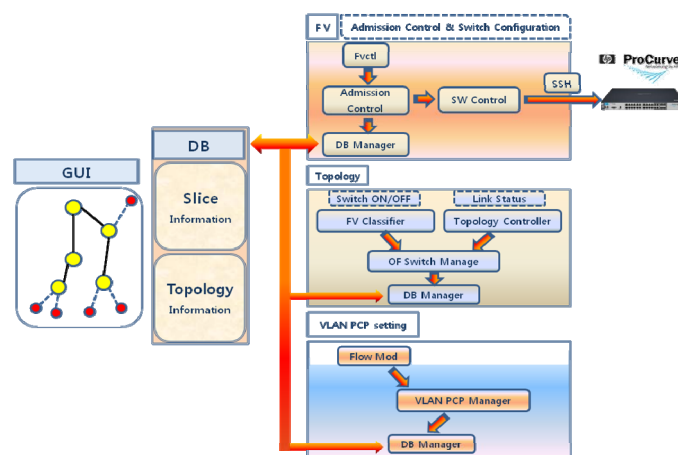


Figure 3. Proposed Architecture for Enhanced FlowVisor

Fig. 3 shows the architecture of our enhanced FlowVisor. First, a database for slice and topology information is implemented. The information in this database is required for admission control and OpenFlow GUI. It also has admission control module, topology control module, and VALN PCP setting module for its QoS control action on top of the basic

FlowVisor function modules. The database structure of the enhanced FlowVisor is shown in Fig. 4.

SliceName	SliceIP	AllocatedBW	VLANPCP	cre_t
1 KISTI-A	tcp:134.75.85	300	1	2011.09.16_14
2 KISTI-B	tcp:134.75.85	200	2	2011.09.16_14
3 KISTI-C	tcp:134.75.85	100	3	2011.09.16_14
4 KISTI-D	tcp:134.75.85	150	4	2011.09.16_14

Figure 4. The database of Enhanced FlowVisor

When a slice creation request is accepted, sum of previously allocated bandwidth is compared to the link bandwidth for each link in the corresponding flow path. If user's request bandwidth is larger than the minimum available bandwidth of all candidate links in the corresponding flow path, then the request is rejected. Otherwise, the request is accepted and a VLAN PCP is assigned to the flow on the created slice and this information is stored into the database. For guaranteeing bandwidth of created slice, it's necessary to configure GMB on the corresponding output ports of switches along the flow path after logging as an administrator.

The command for QoS slice creation, as an example shown in Table 1, is extended by adding required bandwidth at the basic slice creation command. FlowVisor stores this configuration information in xml file and it is used for updating database.

TABLE 1. EXAMPLE FOR QoS SLICE CREATION COMMAND

```
# Slice creation called KISTI-A requiring 300Mbps
/root/fv_test/bin/fvctl --passwd-file=/root/shmin/cnu_passwd
createBWSlice KISTI-A tcp:134.75.85.xxx:6633 minsh@cnu.ac.kr
300M
```

Traffic scheduling for guaranteeing bandwidth is initiated by assigning VLAN PCP field and corresponding bandwidth. The VLAN PCP varies from 0 to 7, and the ratio of total bandwidth is allocated to each PCP field. The sum of assigned ratio does not exceed 1.0. VLAN PCP 0 has lowest priority and is used as a default value. PCP 0 traffic enters into Q3, and PCP 1(2) traffic enters into Q1(Q2). As the PCP value increases, the priority increases. The remaining bandwidth after assigning minimum bandwidth of low priority traffic can be allocated to the high priority traffic. For example, if we assign PCP value 1 to given slice in Fig. 5, it enters into Q1 and 10% of 1Gbps link is allocated. So, even if the sum of traffic coming from other slices exceeds 1Gbps, traffic configured to PCP 1 value can be guaranteed to use 100 Mbps as shown in Fig. 5.

Port	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
1	10	30	10	0	0	0	0	0
2	10	30	10	0	0	0	0	0
3	10	30	10	0	0	0	0	0
4	10	30	10	0	0	0	0	0
5	10	30	10	0	0	0	0	0
6	10	30	10	0	0	0	0	0
7	10	30	10	0	0	0	0	0
8	10	30	10	0	0	0	0	0
9	10	30	10	0	0	0	0	0
10	10	30	10	0	0	0	0	0
11	10	30	10	0	0	0	0	0
12	10	30	10	0	0	0	0	0
13	10	30	10	0	0	0	0	0
14	10	30	10	0	0	0	0	0
15	10	30	10	0	0	0	0	0
16	10	30	10	0	0	0	0	0
17	10	30	10	0	0	0	0	0
18	10	30	10	0	0	0	0	0
19	10	30	10	0	0	0	0	0
20	10	30	10	0	0	0	0	0
21	10	30	10	0	0	0	0	0
22	10	30	10	0	0	0	0	0
23	10	30	10	0	0	0	0	0
24	10	30	10	0	0	0	0	0
A1	10	30	10	0	0	0	0	0
A2	10	30	10	0	0	0	0	0
A3	10	30	10	0	0	0	0	0
A4	10	30	10	0	0	0	0	0

660.0~661.0	sec	11.6	MBytes	97.3	Mbits/sec
661.0~662.0	sec	10.8	MBytes	90.4	Mbits/sec
662.0~663.0	sec	11.5	MBytes	96.1	Mbits/sec
663.0~664.0	sec	11.4	MBytes	95.4	Mbits/sec
664.0~665.0	sec	11.1	MBytes	93.4	Mbits/sec
665.0~666.0	sec	11.0	MBytes	92.5	Mbits/sec
666.0~667.0	sec	11.2	MBytes	94.0	Mbits/sec
667.0~668.0	sec	10.9	MBytes	91.5	Mbits/sec
668.0~669.0	sec	11.2	MBytes	94.0	Mbits/sec
669.0~670.0	sec	11.7	MBytes	97.9	Mbits/sec
670.0~671.0	sec	11.4	MBytes	95.4	Mbits/sec
671.0~672.0	sec	11.4	MBytes	95.4	Mbits/sec
672.0~673.0	sec	11.4	MBytes	95.4	Mbits/sec
673.0~674.0	sec	9.95	MBytes	83.5	Mbits/sec
674.0~675.0	sec	11.4	MBytes	95.4	Mbits/sec
675.0~676.0	sec	11.4	MBytes	95.4	Mbits/sec
676.0~677.0	sec	11.4	MBytes	95.4	Mbits/sec

Figure 5. OpenFlow HP Switch QoS configuration

VLAN PCP is modified when NOX controller sends FlowMod command via Enhanced FlowVisor to HP Procurve switch as follows.

- FlowMod command is received from NOX controller to the enhanced FlowVisor
- The enhanced FlowVisor queries slice information of NOX controller from database
- If there's information about the NOX controller, look up corresponding PCP information
- Extend the FlowMod command to set VLAN PCP as a specified value, 1
- Send the modified FlowMod command from enhanced FlowVisor to Procurve switch.
- Store received information in the flow table at the Procurve switch

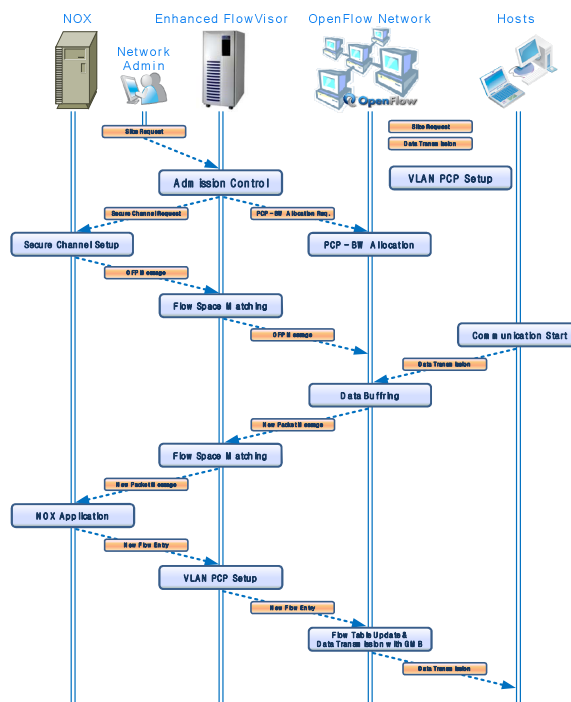


Figure 6. OpenFlow HP Switch QoS configuration

Fig. 6 shows the flow chart of enhanced FlowVisor's procedure from slice creation to packet transmission. When a NOX user requests a slice including bandwidth requirement to the FlowVisor using 'fvctl' command, FlowVisor performs admission control and assigns VLAN PCP and required link bandwidth. After that, FlowVisor sends VLAN PCP and bandwidth assignment message to OpenFlow switch and opens secure channel to NOX controller.

If an OpenFlow switch receives a new flow, then it sends Packet_in message to FlowVisor. FlowVisor sends this message to the NOX controller matching with FlowSpace. NOX controller sends FlowTable to switch via FlowVisor. FlowVisor intercepts this message and attaches Action item for adding VLAN PCP. After that, the OpenFlow switch controls bandwidth based on this PCP value.

IV. PERFORMANCE EVALUATION

To evaluate the enhanced FlowVisor, we deployed a testbed on the KREONET (Korea Research Environment Open Network). The testbed consists of 3 NOX controller and 4 OpenFlow HP Procurve switches. Each switch is located in Seoul, Daejeon A, B and Gwangju. We measured and compared the performance of 2 video streaming slices when background Iperf [10] traffic (3rd slice) shares the bottleneck link with the two video streaming slices as shown in Fig. 7.

The test scenario is as follows. First, non-QoS guaranteed video stream on the slice #0 controlled by KISTI-A controller starts first. Next, UDP Iperf traffic on the slice #1 controlled by KISTI-B starts and traffic generation rate is increased until it approaches the bottleneck link capacity. Lastly, QoS video stream on the slice #2 controlled by KISTI-C starts.

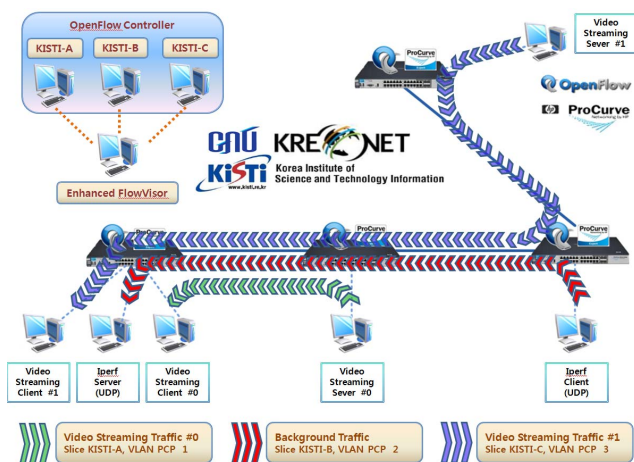


Figure 7. OpenFlow HP Switch QoS configuration

In this experiment, non-QoS video stream is corrupted due to packet loss and delay, but QoS video stream shows guaranteed quality because this traffic receives pre-allocated designated portion of link capacity using GMB setting. The quality of each video stream is shown in Fig. 8.

OpenFlow GUI is implemented by obtaining network information from 'LAVI' at the FlowVisor. Fig. 9 shows the slice topology and flow information on the OpenFlow GUI software.

QoS Non-QoS

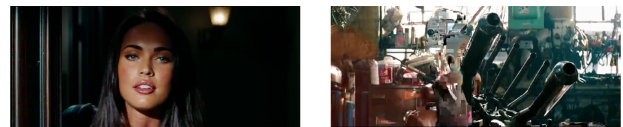


Figure 8. Quality comparison between QoS and non-QoS video stream

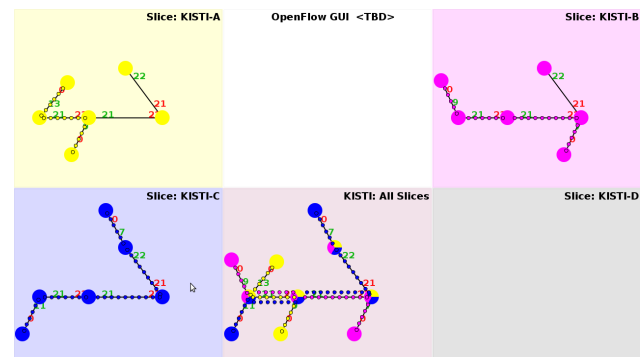


Figure 9. Quality comparison between QoS and non-QoS video stream

V. CONCLUSIONS

In this paper, we implemented enhanced FlowVisor for OpenFlow network virtualization which includes admission control and scheduling module. This is imperative to share testbed while guaranteeing each user's bandwidth requirement. To support this function, admission control and scheduling functions with related database are added to the enhanced FlowVisor. Using this enhanced FlowVisor, video stream on the QoS-guaranteed slice can show perfect quality irrespective of background traffic overflow.

ACKNOWLEDGMENT

This research was partly supported by the Dept. of Computing and Networking Resources of Supercomputing Center, Korea Institute of Science and Technology Information(KISTI) and the project(KCA-2011-09913-05006), "Development of the core technology and virtualized programmable platform for Future Internet". We'd like to express our gratitude for the concerns to spare no support for the research and development of the projects.

REFERENCES

- [1] NSF NeTS FIND Initiative, <http://www.nets-find.net>.
- [2] Future Internet Assembly, <http://www.future-internet.eu/home/future-internet-assembly.html>.
- [3] Shuji Esaki, Akira Kurokawa, and Kimihide Matsumoto, "Overview of the Next Generation Network," NTT Technical Review, Vol.5, No.6, June 2007
- [4] GENI: Global Environment for Network Innovations, <http://www.geni.net/>
- [5] FIRE: Future Internet Research and Experimentation, <http://cor-dis.euro-pa.eu/fp7/ict/fire/>
- [6] OpenFlow forum, <http://www.openflow.org>
- [7] <http://www.wikipedia.org>
- [8] OpenFlow Wiki, <http://www.openflow.org/wk/index.php/FlowVisor>
- [9] NOX forum, <http://noxrepo.org>
- [10] Iperf site, <http://iperf.sourceforge.net/>