Journal of Advances in Shell Programming

Volume 1, Issue 1
www.stmjournals.com

Open Flow Network Visualization Software

***Vikas Bandaru*[1]*, *K. Nikitha*[1], *B. Amrutha Rao*[2]**
[1]GITAM University, Hyderabad, Andhra Pradesh, India
[2]Head Computer Networks Division, RCI, Hyderabad, Andhra Pradesh, India

Abstract
Traditional IP network infrastructure is merely static, rather than flexible. Due to the advent of Cloud technologies, future networks of machine-to-machine (M2M)/Internet of Things (IoT) and the mobile devices generating Big Data, it has become very difficult for network managers to configure and monitor the ever-changing network dynamically. Recently, the concept of Software-Defined Network (SDN), which allows us to administer and configure a network in a centralized and programmable manner, has gathered network engineers' and researchers' attention rapidly. OpenFlow has opened the door for network management in a more controlled way. Though OpenFlow controllers provide a lot of raw data collected from the underlying flows, it is still difficult for network managers to do analysis of that Big Data and take better decisions for the network. This research aims to realize a visualization software that facilitates the network managers to analyze the flow data in a more user-friendly way. The proposed tool helps us to visually identify selected flows from all lists of flows (i.e., Flowspace) and to monitor the status of selected flows through application-specific interactive visual representations.

Keywords: *OpenFlow, software defined network, programmable network, flow visualization, network management, open daylight*

***Author for Correspondence** E-mail: vikas.bandaaru@gmail.com

INTRODUCTION

Computer networks have become part of the critical infrastructure of our businesses, home and schools. Networks are typically built from a large number of network devices such as routers, switches and numerous types of middle boxes (e.g., Firewall) with many complex protocols implemented on them. The behavior of the network depends on the configuration of these thousands of constituent network devices, each of which is configured independently. Today network operators implement high-level network tasks with low-level configuration commands; operators frequently make mistakes when making changes to network configuration [1, 2]. However, there is a lot of research being conducted in the field of programmable networks after the emergence of the concept called software-defined networking (SDN). The designs for next-generation Internet envision inherent flexibility and programmability in the underlying network infrastructure. OpenFlow, as an emerging tool for SDNs, enables network innovations based on commercial switching hardware by separating the intelligent control plane from data-path processing [3]. In general, an OpenFlow network is usually composed of an OpenFlow controller responsible for control functions for handling network packets and multiple OpenFlow-capable network switches. OpenFlow protocol provides an open standard to manage flow-tables in multi-vendor network facilities. These switches and routers communicate with the controller through the OpenFlow protocol. Making use of the protocol, a network administrator can partition traffic into production and research flows. SDN is also being researched for wireless networks [4], emphasizing on wireless personal area networks (WPANs).

In general, the operation of an OpenFlow network requires the development of an OpenFlow controller, so that the OpenFlow controller as a software program can handle flow-tables on OpenFlow-enabled network switches and routers [5]. For this reason, several kinds of development frameworks which facilitate us to code OpenFlow controller, such as POX [6], Trema [7], Floodlight [8] and OpenDaylight [9] have been available.

Despite the existence of such frameworks, the developer tends to have the following difficulty. The difficulty lies in understanding the topology of the entire OpenFlow network, actual paths of the network flows, traffic amount on each link of the network, types of services/applications generating the traffic, the amount of traffic being consumed by any specific service/application on any specific device and so forth.

Flow visualization, which gathers and shows the information and behavior (e.g., network topology, traffic statistics, network and device configuration parameters, and others) of all the flows in a network, extensively provides useful insight about the underlying network. Especially, flow visualization can help us perform network management decisions based on monitored network information.

To understand each flow better, we also need to consider the related service(s) with their applications. In this paper, the authors propose an attempt, called as *application-specific interactive flow visualization* in OpenFlow-based programmable networks, which can interactively map the flows onto applications. The proposed tool helps us to visually identify (i.e., separate) selected flows from all lists of flows (i.e., Flowspace) and to monitor the status of selected flows. That is, in order to visualize flows with application mapping, we attempt to classify a variety of Internet applications/services into distinctive types and to analyze each flow with the classified distinctive types.

In this paper is reported a research work in progress towards OpenFlow network visualization software, which can facilitate to efficiently control and monitor an OpenFlow network. The rest of the paper is structured as follows. In section Problem In Current OpenFlow Visualization Software is reviewed the issues with current OpenFlow visualization software. In section Proposal of Our Work in Progress is shown the OpenFlow network visualization software which allows us to view the topology of our target OpenFlow network and operate flows on it with ease. The results are concluded in section Conclusions.

PROBLEM CURRENT OPENFLOW VISUALIZATION SOFTWARE

Gathering real-time statistics about the network state through visualization can help us perform network management decisions and monitor network-related problems. Several visualization systems have been designed in the context of OpenFlow; however, such systems are constrained by their fundamental limitations. For example, LAVI [10] and NOX GUI [11] have restricted towards visualizing the network that are connected to a NOX controller. Both systems lack generality to work with other controllers. ENVI [12], the frontend unit, communicates with LAVI by requiring an intermediate binary to JSON format translator. Such intermediate level of abstraction requires significant development time and making it hard to port when extended to work with other OpenFlow-based controllers.

ROVIZ [13] took the above problem into consideration and contributed the following solutions: (i) compatible with any OpenFlow-based network, irrespective of the controller(s), (ii) maintains real-time information about the entire network as well as individual devices, and (iii) provides interactive menus for selective information retrieval improving bandwidth constraints.
Application-aware aggregation, which just depends on port-to-application mapping, is time-consuming only with manual detection for application awareness [14]. This is in some extent solved by Shin and Kim [15] where the services are subcategorized into three types: (i) static port-based type, (ii) dynamic port-based type, and (iii) unknown type. Even though Shin and Kim [15] solved the problem to an extent, it is not sufficient to monitor the network and its traffic usage based on per application on individual devices.

PROPOSAL OF OUR WORK IN PROGRESS

For the consideration above, the authors have been prototyping visualization software that can give OpenFlow network managers an intuitive understanding of how network flows are formed, what the topology of a physical OpenFlow network is like, how much bandwidth is being used by which applications in which devices, dynamically add and remove flow entries directly to the OpenFlow switches via OpenFlow controller for experimentation purposes with the implementation of flow control interface [16].

Figure 1 overviews the architecture of the proposed visualization software. The visualization software under development is composed of visualization module, communication module built on top of an OpenFlow controller, and OpenFlow switches. The visualization module is in charge of a User interface (UI) to the network manager, visualizing a topology of the OpenFlow network as a graph and then superimposing network flow information on the graph. Also, through the visualization module, the managers can add and remove flow entries to OpenFlow switches. The communication module built on OpenFlow controller is responsible for obtaining the information on network from controller and mediating the interaction between the manager UI and OpenFlow switches. For the communication between the communication module and OpenFlow switches, OpenFlow protocol is used. For the development of visualization tool, authors have adopted Open Daylight, which is a Java based OpenFlow controller, focusing on the characteristics of enabling the development of OpenFlow visualization tool in Java.

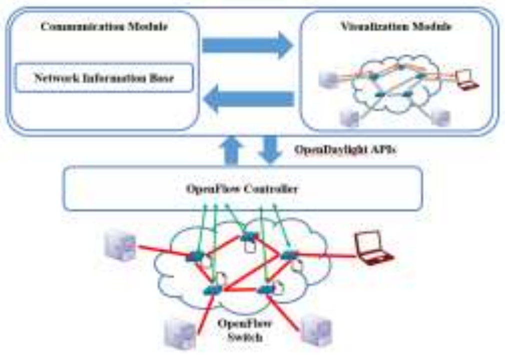


Fig. 1: *Overview of the Proposed Architecture.*

A. Visualization Module

The visualization module has three functionalities. The first functionality is to visualize the topology of the target OpenFlow network and superimpose the information on network flows on it. The information on network flows includes traffic amount on each link between OpenFlow switches composing the OpenFlow network, a list of network flows, and individual flow information as and when a specific flow is selected. The second functionality is to show the graphical representation of bandwidth usage and traffic flow between devices as per application. For visualization of network, the authors have adopted Jung 2.0.1 (Java Universal Network/Graph Framework) [17]. The third functionality is to allow the developer to directly describe flow entry parameters and add and remove flows. Figure 2 shows the prototype diagram of the visualization software under development.

B. Communication Module

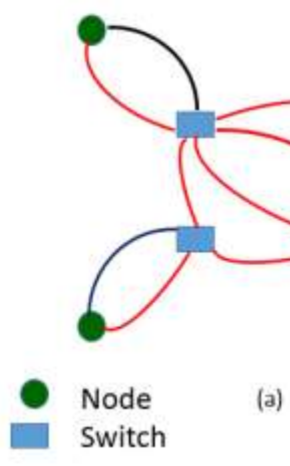
The communication module consists of *Network Information Base*. Network information base collects the topology, flow and link information from the underlying network switches through the OpenDaylight controller and communicates the same to visualization module for graphical representation. It makes the topology graph database of the OpenFlow network devices by utilizing LLDP (link layer discovery protocol), a protocol for information exchange between neighbor devices. The communication module first instructs each OpenFlow switch to send LLDP packets to all neighbors. A LLDP packet contains the ID of the sender switch and the ID of the port that it is sent through. On receiving the LLDP packet from a neighbor switch, OpenFlow switches report its arrival to the communication module. With this mechanism, the communication module can obtain the information on all links and then build the topology graph of the OpenFlow network which OpenFlow switches form. Furthermore, to understand the connections between OpenFlow switches and hosts, the communication module works as follows. In the event that a new host is connected to an OpenFlow switch and sends out any packet, the event is informed to the communication module built on the OpenFlow controller. Then, the communication module extracts source MAC address and IP address contained in the packet. If their address is unknown to the communication module, it adds the corresponding connection between the OpenFlow switch and the new host to the graph obtained by the LLDP-based mechanism above.

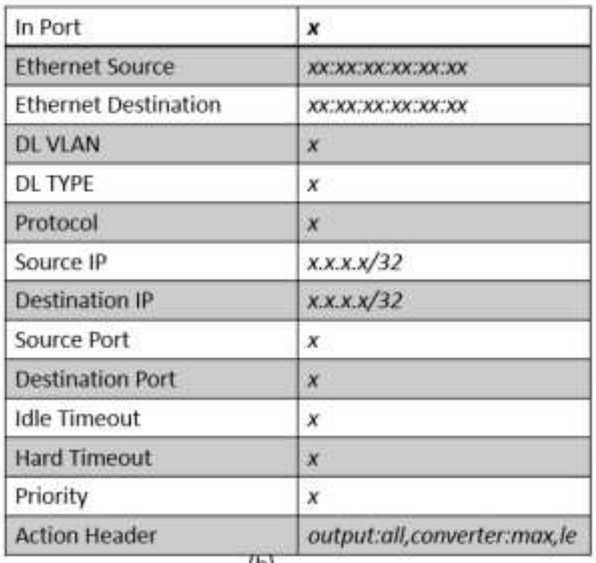
To obtain the information on network flows on the OpenFlow network, the communication module analyzes flow entry information retrieved from each OpenFlow switch. The flow entry information retrieved from an OpenFlow switch contains only which port to send out for a series of packets that matches a rule set in advance. Furthermore, the information is fragmentary and thus the developer has difficulty in understanding the entire network flows on the target OpenFlow network. In the communication module, “match,” “actions,” and “packet count” are focused. In order to understand the network flows on the OpenFlow network, the communication module obtains the information from each switch every one second. Also, the traffic amount of each network flow is estimated from the comparison of “byte count” flowing on a specific port between the information obtained every one second.

The communication module also has the ability to modify the actual path of a specified flow. This is conducted by adding new flow entries which override the flow entries of the specified flow. Note that the modified path will be reverted some time later because the OpenFlow controller will refresh the flow entries in the event of timeout.

C. OpenFlow Switch

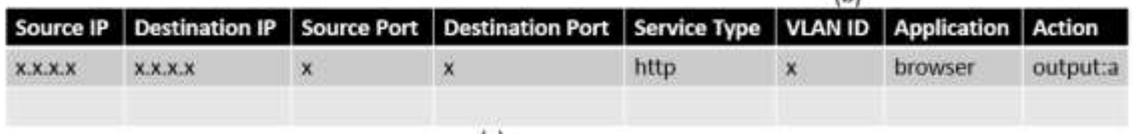
As of writing this paper, we do not have enough OpenFlow switches to make an OpenFlow network. In this research, Open vSwitch [18], [19] has been used for emulation of OpenFlow switch. Open vSwitch is a software implementation of an OpenFlow switch intended to run as a virtual switch in virtualized environments. Open vSwitch can operate both as a software switch running within the hypervisor, and as the control stack for switching silicon. It has been ported to multiple virtualization platforms and switching chipsets. In this research, an OpenFlow network has been constructed with Open vSwitches on a Linux machine.





In Port	*x*
Ethernet Source	*xx:xx:xx:xx:xx:xx*
Ethernet Destination	*xx:xx:xx:xx:xx:xx*
DL VLAN	*x*
DL TYPE	*x*
Protocol	*x*
Source IP	*x.x.x.x/32*
Destination IP	*x.x.x.x/32*
Source Port	*x*
Destination Port	*x*
Idle Timeout	*x*
Hard Timeout	*x*
Priority	*x*
Action Header	*output:all,converter:max,le*

(b)



Source IP	Destination IP	Source Port	Destination Port	Service Type	VLAN ID	Application	Action
x.x.x.x	x.x.x.x	x	x	http	x	browser	output:a

(c)

Fig. 2: *(a) Traffic Flow between Switches and Nodes.*
(b) Add New Flow. (c) Sample Table in Database.

CONCLUSIONS

In this paper, the authors' research work in progress toward OpenFlow network visualization software was reported. The authors have focused on the difficulties in developing an OpenFlow visualization tool and thus started the development of OpenFlow network visualization software for the purpose of facilitating the managers' monitoring of OpenFlow network. At the time of writing this paper, the software is still under development. The s would like to improve the prototyped visualization software obtaining the feedback from actual developers and engineers working around OpenFlow network.

REFERENCES

1. Feamster N, Balakrishnan H. Detecting BGP Configuration Faults with Static Analysis. *Proceedings of 2nd USENIX NSDI,* Boston, MA, May, 2005.
2. Mahajan R, Wetherall D, Anderson T. Understanding BGP misconfiguration. *ACM SIGCOMM*, Pittsburgh, PA; Aug. 2002; 3–17p.
3. McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: Enabling innovation in campus networks. *White Paper*. March 2008.
4. Salvatore Costanzo, Laura Galluccio, Giacomo Morabito, et al. Software defined wireless networks: unbridling SDNs. *IEEE*. 2012.
5. Furuichi T, Date S, Yamanaka H, et al. A prototype of network failure avoidance functionality for SAGE using OpenFlow. *IEEE 36th International Conference on Computer Software and Applications Workshops* (The Sixth Middleware Architecture in the Internet (MidArch 2012)). Jul. 2012; 88–93p.
6. POX, http://www.noxrepo.org/pox/about-pox/.
7. Trema, http://trema.github.io/trema/.
8. Floodlight OpenFlow Controller – Project Floodlight, http://www.projectfloodlight.org/floodlight/.
9. Project OpenDaylight, http://www.opendaylight.org/
10. Yap KK. *Network Visualization.* http://www.openflowswitch.org/wk/index.php/LAVI, May 2009.
11. Zarifis K. Nox gui. http://noxrepo.org/nox wiki/index.php/NOX-GUI, October 2010.
12. Underhill DG. An extensible network visualization and control framework. *Thesis*. Stanford University, May 2009.
13. Natarajan S, Huang X. An interactive visualization framework for next generation networks. *ACM CoNEXT*. Nov, 2010.
14. Das S, Yakoumis Y, Parulkar G, et al. Application-aware aggregation and traffic engineering in a converged packet-circuit network. *OFC/NFOEC,* March 2010.
15. Shin Sungho; Kim JongWon. Toward service-aware flow visualization over openflow-based programmable networks. *Asia-Pacific Advance Network*. 2011; *v*. 32: 8–13p.
16. Watashiba Y, et al. OpenFlow Network Visualization Software with Flow Control interface. *Computer Software and Applications Conference, IEEE*. 2013.
17. JUNG-Java Universal Network/Graph Framework, http://jung.sourceforge.net/.
18. Pfaff B, Pettit J, Koponen T, et al. Extending networking into the virtualization layer. HotNets-VIII. 2009.
19. Open vSwitch, http://openvswitch.org/.