J Inf Process Syst, Vol.12, No.3, pp.511~524, September 2016
http://dx.doi.org/10.3745/JIPS.03.0039

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

# SDN-Based Enterprise and Campus Networks: A Case of VLAN Management

Van-Giang Nguyen* and Young-Han Kim*

**Abstract**

The Virtual Local Area Network (VLAN) has been used for a long time in campus and enterprise networks as the most popular network virtualization solution. Due to the benefits and advantages achieved by using VLAN, network operators and administrators have been using it for constructing their networks up until now and have even extended it to manage the networking in a cloud computing system. However, their configuration is a complex, tedious, time-consuming, and error-prone process. Since Software Defined Networking (SDN) features the centralized network management and network programmability, it is a promising solution for handling the aforementioned challenges in VLAN management. In this paper, we first introduce a new architecture for campus and enterprise networks by leveraging SDN and OpenFlow. Next, we have designed and implemented an application for easily managing and flexibly troubleshooting the VLANs in this architecture. This application supports both static VLAN and dynamic VLAN configurations. In addition, we discuss the hybrid-mode operation where the packet processing is involved by both the OpenFlow control plane and the traditional control plane. By deploying a real test-bed prototype, we illustrate how our system works and then evaluate the network latency in dynamic VLAN operation.

**Keywords**

Campus Network, Enterprise Network, OpenFlow, Software Defined Networking (SDN), VLAN Management

## 1. Introduction

The concept of a Virtual Local Area Network (VLAN) was used for the first time around 30 years ago. Even though it has a long history, it is still used as one of the most popular network virtualization technologies in today's enterprise and campus networks. The VLAN in enterprise and campus networks is commonly used to group hosts in administrative domains and disregards their physical locations in the network topology. Example groups are engineers, sales, student clusters, managers, faculty clusters, etc. By doing so, network administrators can reduce the complexity of their management tasks, as well as provide better security and reduce costs. For example, VLAN can help network administrators create many smaller broadcast domains from a large on instead of using expensive routers. As surveyed in [1], the authors stated that VLAN is a useful mechanism for limiting the scope of broadcast traffic, enforcing security and privacy policies, simplifying access control, decentralizing network management, and enabling host mobility.


Manuscript received October 22, 2014; accepted October 22, 2015; onlinefirst December 7, 2015.
**Corresponding Author:** Young-Han Kim (yhkim@dcn.ssu.ac.kr)
* School of Electronic Engineering, Soongsil University, Seoul, Korea ({nvgiang, yhkim}@dcn.ssu.ac.kr)

In general, static VLAN and dynamic VLAN are the two most common ways of configuring VLAN. In a static VLAN, the network administrator manually creates a VLAN at a switch and then assigns its ports to the proper VLAN through the command line interface (CLI). Therefore, the static VLAN configuration is fairly straightforward and is a so-called port-based VLAN. A dynamic VLAN, on the other hand, can automatically assign the VLAN for a user using mapping information between devices (e.g., MAC address) and a VLAN ID. This mapping information is received from a dedicated server called a VMPS (VLAN Member Policy Server).

However, VLAN configuration remains a tedious, complex, and error-prone process because network administrators need to manually configure a lot of individual network devices (i.e., switches) by using device-level configuration CLI. It would be more difficult if these devices were located in a larger location (e.g., different floors of a big building), which means that it is difficult for network administrators to scale to the scope of the entire network. Therefore, VLAN management is one of the most challenging tasks that network administrators face. Despite the fact that Cisco has already supported a new protocol named the VLAN Trunking Protocol (VTP) [2] to better manage their devices, it is still very limited in terms of scope and functionality. Indeed, it requires creating VTP domains so that VTP advertisements can be transferred. In addition, VTP is only available on Cisco switches, such as Catalyst Family switches. We believe that to deal with these issues, there is a need for a VLAN management tool or an application that not only allows network administrators to easily configure VLAN but also that provides a virtual view for monitoring and troubleshooting each VLAN in the network.

Recently, Software Defined Networking (SDN) [3] has emerged as one of the most attractive technologies in networking research. It features the separation of the control plane and the data plane, and network programmability. The control plane is composed of various network applications that decide how the traffic is traversed, while the data plane consists of simple forwarding devices controlled by an SDN controller via programmable interfaces (e.g., OpenFlow [4]). By doing so, SDN simplifies network management and configuration because administrators do not need to do some of the device-level configuration tasks. Moreover, the complex control logic can be defined by writing programs, which is much easier to implement, upgrade, and maintain.

OpenFlow [4] was proposed as a standard protocol between the switches and the software-based controller in SDN architecture. Through OpenFlow protocol, the OpenFlow controller maintains and updates flow entries in switches' flow tables. In order to do so, the OpenFlow controller can perform two flow management approaches: proactive and reactive. In the proactive approach, all required flow entries are pre-installed in the flow table in each switch by the controller. In contrast, the reactive approach requires the controller to process every unknown packet coming into the network. Due to the numerous benefits of SDN and OpenFlow, network operators have been adopting SDN and OpenFlow into many aspects of the network, such as campus and enterprise networks, the carrier network, and the mobile network [5].

Although SDN and OpenFlow can replace all of the management models of current network architectures, we understand that the usage of VLAN is still needed. One reason is that network administrators who are operating the traditional enterprise and campus network somehow want to keep the use of VLAN for constructing their virtual networks. To address the aforementioned issues of VLAN management in enterprise and campus networks, SDN and OpenFlow would be a good solution. With OpenFlow and SDN we can easily add a tag to the traffic and remove it after finishing via an

action from the SDN controller. The objective of this paper is to develop an OpenFlow application that can be used to help the network administrator configure and manage the VLAN in the network more flexibly and efficiently.

The main contributions of our work are as listed below:

- Introduce a new architecture for enterprise and campus networks by adopting a SDN and OpenFlow concept.
- Design and develop an application for managing VLAN for this architecture that supports both static VLAN and dynamic VLAN configurations.
- Provide an interactive graphical user interface (GUI) for virtualizing and monitoring the whole network and VLAN network. Implement a prototype testbed to demonstrate our system using a Floodlight controller [6] in both an emulation testbed based on a Mininet environment [7] and a real testbed comprised of OpenFlow-enabled embedded-hardware switches.

The remainder of the paper is organized as follows: we first review some related works in Section 2. In Section 3, we present an enterprise and campus network based on SDN and OpenFlow. The detailed design and implementation of our VLAN management application are described in Section 4. The evaluation and results are presented in Section 5 and in Section 6, we conclude our paper.

## 2. Related Works

Before the advent of SDN and OpenFlow, Krothapalli et al. [8,9] proposed a virtual MAN, which is a VLAN management system for managing VLAN in traditional enterprise networks. The objective of this work is to provide a tool for automating and visualizing VLAN configurations. It also assists network administrators in detecting and troubleshooting VLAN misconfigurations, such as missing and redundant links, through an interactive graphic interface.

Considering the hybrid deployment of OpenFlow and SDN in the traditional enterprise network, several works [10,11] have been proposed. HybNET [10] proposed a framework for the automated network management of hybrid OpenFlow/SDN-legacy networks. One of this work's contributions is that it provides network virtualization functionality for a set of hosts by using VLAN. Panopticon [11] presented the design and implementations of a large enterprise network architecture that combines legacy and SDN switches. In order to establish the transition between SDN and legacy networks, the authors proposed solitary confinement tree (SCT) structures where VLAN IDs are used to steer the traffic headed for the legacy switch's ports towards SDN switches. Our approach also considers the hybrid mode for VLAN management, as described in Section 4.

The works closest to ours are Ref [12-14]. The authors in [12] proposed a framework using OpenFlow to manage the VLANs of an eduroam network. In this approach, they used another parameter called virtual Group IDs (GIDs) for creating the virtual network, instead of VLAN IDs. By doing so, VLAN's limitations are overcome. However, we argue that 4,096 VLAN IDs seem to be enough for creating VLAN networks, while it can be reused. A. Lara et al. [13] proposed a solution to overcome two challenges in network management, including encryption control and traffic isolation in enterprise networks. They isolated the traffic in the network by using VLAN with the assistance of an OpenFlow controller. However, compared to our system, they did not allow specific VLAN packets to a specific port; instead they used a 'normal' action in flow entries. This means that the packet after being

tagged with a VLAN ID will be further processed by normal operation of the switch. Tsai et al. [14] proposed a mechanism for supporting VLAN tag-translation across wide-area OpenFlow networks in Taiwan. This mechanism allows for building the path between different OpenFlow networks by using two types of VLAN tags: inside and outside VLAN tags.

Sherwood et al. [15] proposed FlowVisor, which is a hypervisor that sits between controllers and switches in OpenFlow networks. FlowVisor allows an administrator to slice an underlying OpenFlow network into several isolated slices or virtual networks. Our system can easily adapt in FlowVisor-based network architecture. By doing so, the network administrator can divide each slice into smaller groups of VLAN networks in a flexible manner.

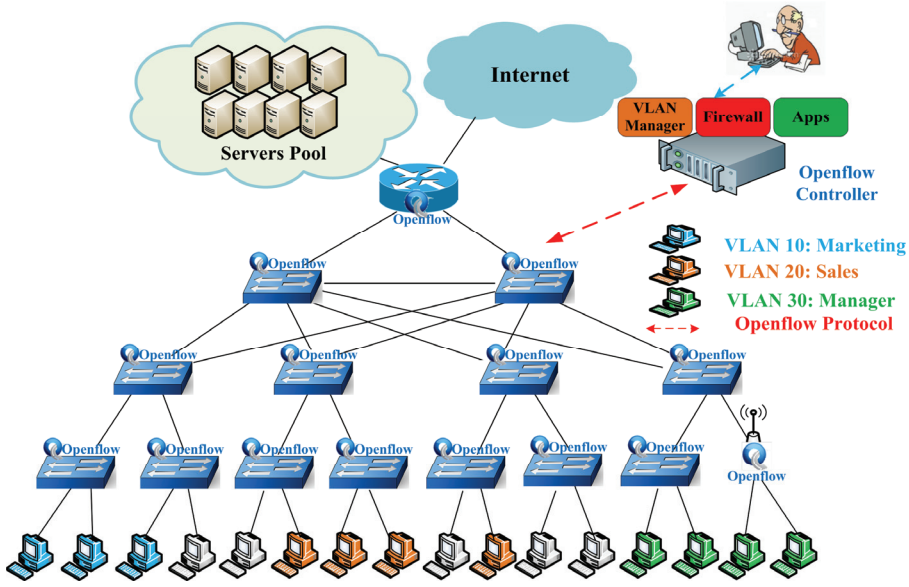


**Fig. 1.** Virtual Local Area Network (VLAN) management in Software Defined Networking (SDN)-based enterprise and campus network.

## 3. SDN-Based Enterprise and Campus Network Architecture

Fig. 1 depicts the architecture of an enterprise and campus network based on SDN and OpenFlow. In this architecture, OpenFlow-enabled devices, as shown in Fig. 2, replace all of the traditional network devices, such as switches, hubs, access points, and routers. Some hardware-based devices, such as the firewall or load balancer, are removed and simplified as applications on the top of an OpenFlow controller. The VLAN manager is an application that allows network administrators to configure VLANs.

Compared with traditional enterprise and campus networks, introducing SDN and OpenFlow can simplify the network while offering significantly greater flexibility as follows:

- Improve network efficiency through centralized management and control along with a higher degree of orchestration and automation. The controller can set policies and provide access control in a dynamic manner.

- Improve service availability, since the controller can compute backup paths in case failure occurs and improve the convergence time compared to the traditional network.
- Provide advanced analytics of all resources so that a business can easily monitor and control these resources and make strategic business decisions efficiently. As such, the operating expenses (OPEX) can be reduced.
- Since services and applications are virtualized, the network operators can reduce their capital expenditure (CAPEX), as well as maximize the utilization of resources.
- In addition, SDN and OpenFlow can simplify the network by freeing it from the use of middleboxes and integrating the middleboxes' functionalities within the controller. Some examples of middlebox functionalities that have been implemented using SDN include the NAT, firewall, load balancer, network access control, etc.

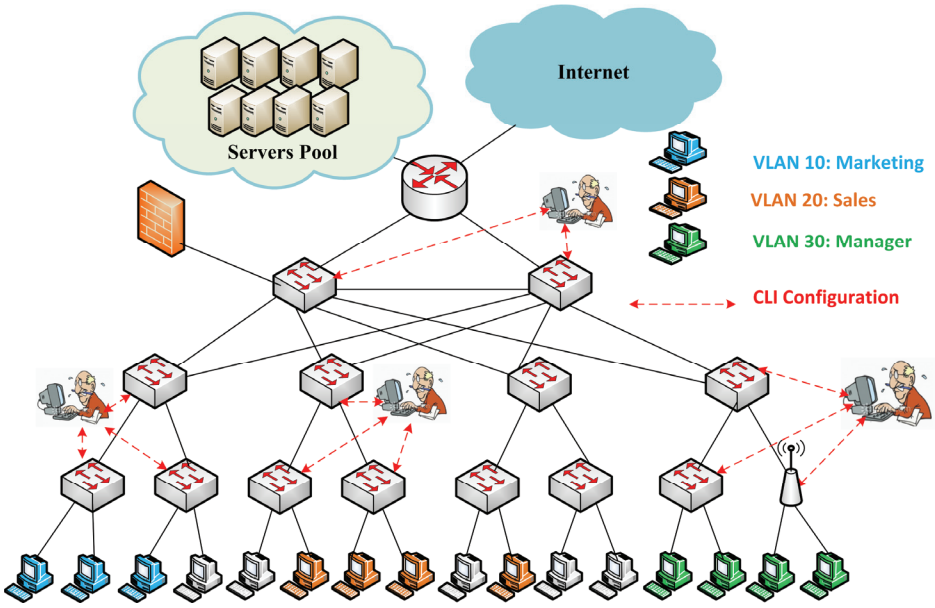


**Fig. 2.** Virtual Local Area Network (VLAN) management in the traditional enterprise and campus network.

## 4. Design and Implementation

In this section, the detailed design of the VLAN manager application is described. This application is built on top of the Floodlight controller [4], which is one of the most popular OpenFlow controllers that have been used in the community. More details are discussed in the subsections below.

### 4.1 Floodlight Controller Overview

The Floodlight controller is an enterprise-class, Apache-licensed OpenFlow controller. It was developed in Java and is now supported by a community of developers, including a number of engineers from Big Switch Networks. It has been tested using both physical and virtual OpenFlow-

compatible switches. It not only can handle an OpenFlow network, but it can also manage multiple "islands" of OpenFlow hardware switches, including non-OpenFlow networks (e.g., traditional L2-switch networks) with high-performance. The Floodlight controller has a modular architecture where we can easily add our own modules and fit them with other. The detailed modules of Floodlight controller architecture are depicted in Fig. 3.

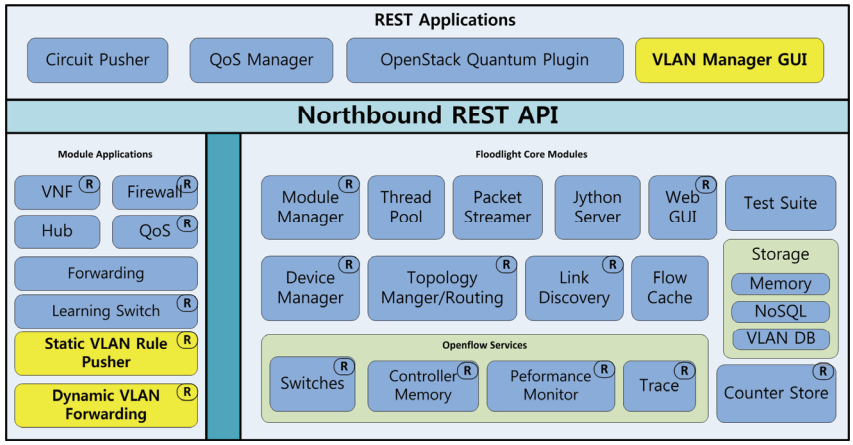


**Fig. 3.** Floodlight controller with new functions depicted in three yellow boxes.

## 4.2 Static VLAN Management

**H1 at port 1 Openflow Switch 01 (DPID=00:0a:6c:3b:e5:f9:7c:80)**

(1) "switch": "00:0a:6c:3b:e5:f9:7c:80","name":"VLAN 10-1","ingress-port":"1","actions":"set-vlan-id=10,output=2"

(2) "switch": "00:0a:6c:3b:e5:f9:7c:80","name":"VLAN 10-2","ingress-port":"2", "vlan-id":"10","actions":"strip-vlan,output=1"

**Intermediate Openflow Switch 02 (DPID=00:00:23:20:e5:f9:7c:2d)**

(3) "switch": "00:00:23:20:e5:f9:7c:2d","name":"VLAN 10-3", "ingress-port":"4","vlan-id":"10","actions":"output=5"

(4) "switch": "00:00:23:20:e5:f9:7c:2d","name":"VLAN 10-4", "ingress-port":"5","vlan-id":"10","actions":"output=4"

**H2 at port 2 of Openflow Switch 03 (DPID = 00:0a:6c:3b:e5:f9:99:74)**

(5) "switch": "00:0a:6c:3b:e5:f9:99:74","name":"VLAN 10-5","ingress-port":"2","actions":"set-vlan-id=10,output=3"

(6) "switch": "00:0a:6c:3b:e5:f9:99:74","name":"VLAN 10-6","ingress-port":"3", "vlan-id":"10","actions":"strip-vlan,output=2"

**Fig. 4.** Example of static Virtual Local Area Network (VLAN) rules added to Openflow switches.

In order to support static VLAN configuration, we have developed a new module, which is called a Static VLAN Rule Pusher, in the Floodlight controller, as depicted in Fig. 3. This module is in charge of all of the static VLAN management tasks. It defines the functions for flexibly configuring and managing the VLAN network, including adding a new VLAN network, deleting a VLAN or all VLANs, editing an existing VLAN, and viewing the VLAN network. Another module is the VLAN Manager GUI, which is a web browser-based GUI that interacts with the Floodlight Core module and Static VLAN Rule Pusher module by using the REST API described above. According to the operation of OpenFlow, we used the

proactive mode for handling flows in static VLAN management. We describe in detail the functions of static VLAN management below.

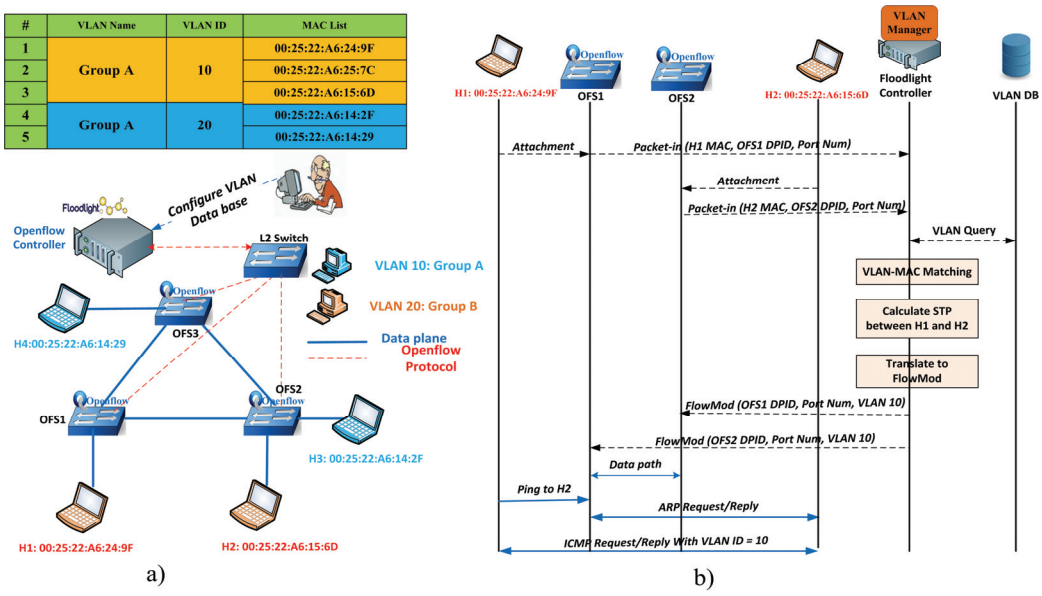


**Fig. 5.** Dynamic Virtual Local Area Network (VLAN) management. (a) Example topology. (b) Procedure for adding VLAN between two hosts.

**Add VLAN**: this function is used to create a VLAN network for a group of hosts. In order to do so, it installs a set of flow rules into related switches with a given VLAN ID. The input information needed is the VLAN name, switch Data Path ID (DPID), VLAN ID, and switch ports, which the VLAN hosts attach to. Some other options can be a priority or the time out for each VLAN, but we will not describe these in more detail in this paper. As for the context of VLAN, we also defined two types of switch ports: tagged and untagged ports. Tagged ports are ports where the packet will be added with a VLAN ID into its header when traversing through. Untagged ports are ports where the VLAN ID will be removed from the packet's header. Forwarding ports are the switch's ports that do not have a VLAN host, but are involved in the VLAN path. Forwarding ports will forward a packet to outgoing ports without modifying anything in its header (i.e., without adding/removing the VLAN ID). Normally, each communication between a pair of hosts consists of two flow entries, one to send the packet in one direction and the other in the reverse direction of the communication. In order to perform VLAN configuration, we used multiple actions to set up flow entries with two key actions, which are namely the set-vlan-id and strip-vlan. Depending on whether the packet is going to a tagged port or untagged port, two of these actions will be set properly. Other actions simply forward the packet to corresponding ports. An example of flow entries is shown in Fig. 4. These flow entries (1)–(6) are used to establish a VLAN segment between two hosts in a 3-OpenFlow switch network. The name of these flow entries are made from the VLAN name and an incremental number so that there is no coincidence between two flow entries. Obviously, we can recognize that port number 1 of OpenFlow switch 01 and port number 2 of OpenFlow switch 03 are tagged ports while port number 2 of OpenFlow switch 01 and port number 3 of OpenFlow switch 03 are untagged ports. OpenFlow switch 02 is an intermediate switch with

forwarding ports, as described above. It simply forwards a packet that matches against with a particular VLAN ID.

**Delete VLAN**: this function is used to delete a VLAN network. It allows the network administrator to remove the VLAN network easily with one click on the web GUI, instead of having to remove the configuration file manually in individual switches. By doing so, all of the flow entries in all of the switches belonging to the VLAN are deleted at the same time.

**Edit VLAN**: this function allows network administrators to modify an existing VLAN segment. Some modifications can add or remove a new host to and from the existing VLAN.

**View VLAN**: This function is used to display all VLAN networks as well as their information. It also provides a separated view for displaying several VLAN network topologies simultaneously.

## 4.3 Dynamic VLAN Management: A Case of VLAN Between a Pair of Hosts

In this section, we propose the method for supporting dynamic VLAN in SDN-based enterprise and campus networks. In the example shown in Fig. 5, two hosts are waiting to be configured in the same VLAN network (with VLAN ID = 10), and when these hosts move, they still continue their connection in this VLAN without being reconfigured by the network administrator. The operation procedure is detailed as shown in Fig. 5 and is described in the steps laid out below. Like the traditional dynamic VLAN, we also used the hosts' MAC address as the VLAN membership parameter. According to the operation of OpenFlow, we used the reactive mode for processing flows in dynamic VLAN management.

We describe step by step below how to set up the VLAN between a pair of hosts.

**Step 1**: Registration

When two hosts want to join to the network with an isolated VLAN network, they will register with the network administrator with their MAC addresses and desired VLAN ID. For instance, in this case, the VLAN ID is 10 and the VLAN network name is Group A.

**Step 2**: Attachment

After successfully registering, they attach to the network. When they first connect to the network, the OpenFlow switches will encapsulate a Packet-in, including the attachment point information (host's MAC address, switch DPID, and the port that the host attaches to), and will send to the Floodlight controller.

**Step 3**: VLAN Matching

Upon receiving the attachment point information, the Floodlight controller sends a query to the VLAN database and makes a match decision between the attachment point information and the VLAN table in the database. The result is a new tupe (attachment point, VLAN ID).

**Step 4**: Path Calculation and Rule Installation

Simultaneously, by using the attachment point information, the Floodlight controller executes the Dynamic VLAN Forwarding module, as shown in Fig. 3. This module performs two operations. First, it calculates the shortest path between the two attachment points by using Dijkstra or Floyd-Warshall algorithms. Then, the DPID list of all of the switches in the shortest path is returned. Finally, by using the information gained from the VLAN matching step, the Floodlight controller installs the rules to the above list's switches by sending FlowMod messages, including the switches' DPIDs, port number, and corresponding VLAN ID.

**Step 5**: Communication

After finishing the rules setup step, the two hosts are able to communicate with each other in an isolated network with the VLAN ID =10.

In case the network administrator wants to remove a VLAN network, he/she can remove all of the information about this VLAN from the VLAN database and can send a command to the controller so that the flow entries of all related switches are removed.

## 4.4 Hybrid Support

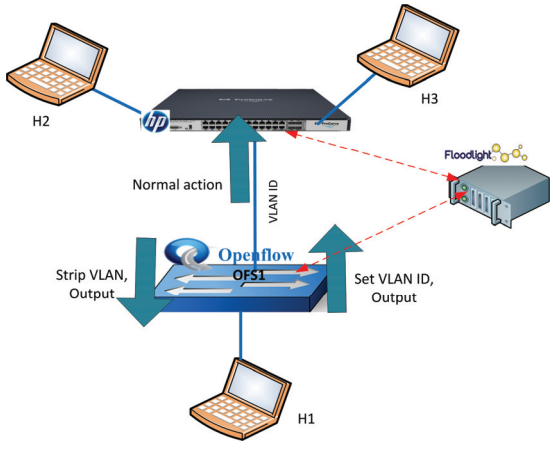


**Fig. 6.** Example of Virtual Local Area Network (VLAN) configuration in hybrid mode.

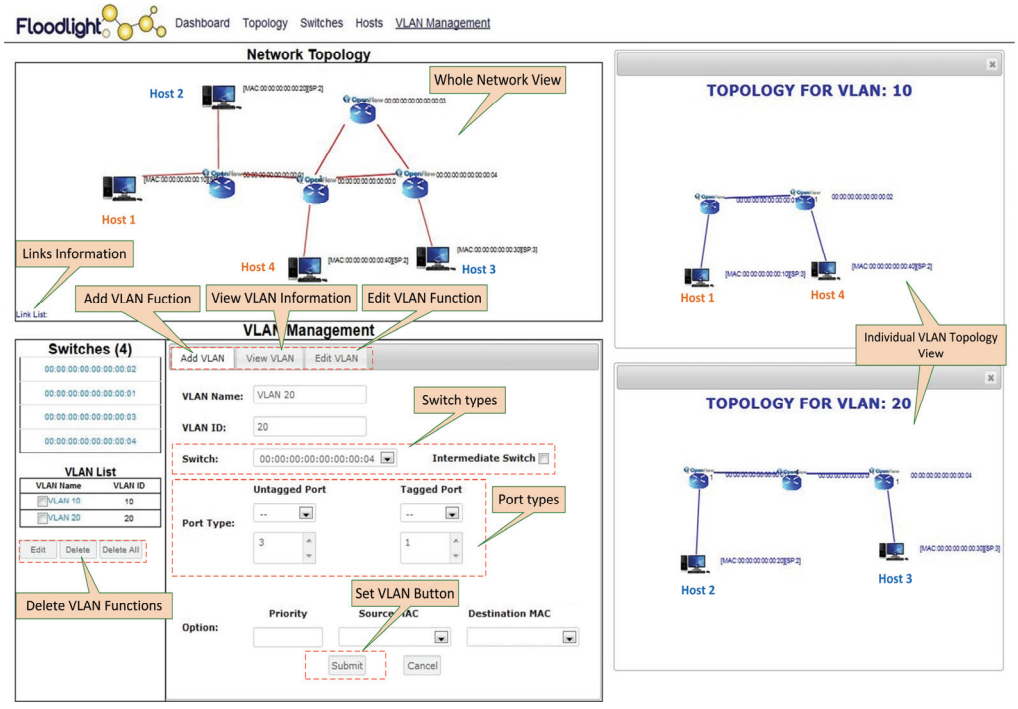


**Fig. 7.** Virtual Local Area Network (VLAN) manager graphical user interface.

In the two scenarios listed above, we assumed that all of the OpenFlow switches used in the network are pure-OpenFlow switches. However, hybrid-mode OpenFlow is one of the possible solutions that is deployed in a real environment and supported by some network vendors' hardware switches. 'Hybrid' means that it can be used to combine the packet processing by the OpenFlow control plane and the traditional distributed control plane. In order to do so, OpenFlow allows for a normal action, which is used as an output port and it processes the packet using the normal pipeline. Consider the example given in Fig. 6 where we attempt to configure the VLAN for three hosts with one HP OpenFlow switch and one pure-OpenFlow switch. Two hosts, H2 and H3, are set up as VLAN by using the traditional CLI method and the host H1 is set up as VLAN by using the two actions of set-vlan-id and strip-vlan for two directions of communication, as shown in Fig. 6. The packet from OFS1 to the HP switch with appropriate VLAN ID will be executed by the normal action. Therefore, the packet is processed and forwarded to the VLAN ports by using the traditional processing pipeline.

## 5. Evaluation and Results

This section describes our initial results. In brief, we already developed a VLAN manager for static VLAN configuration, including a GUI for displaying VLAN network topology. In addition, we evaluated the packet's processing delay for creating a VLAN between a pair of hosts in various scenarios in Mininet [7]. We implemented a prototype system based on the Soekris-based OpenFlow switches, which were built from the OpenvSwitch v1.10. We used the OpenFlow v1.0 in our system.

### 5.1 VLAN Graphic User Interface


**Fig. 8.** A testbed prototype.

The GUI for static VLAN management is depicted in Fig. 7. This GUI consists of three main parts: Network Topology, VLAN Management, and Individual VLAN Topology View. VLAN management has three sub-tabs for performing three functions: Add VLAN, View VLAN, and Edit VLAN. These functions have already been described in Section 4. The Add VLAN tab contains all of the necessary input for setting a new VLAN. The network administrator fills the VLAN name and VLAN ID, while the switches' DPIDs and port number are easier to select. The view VLAN tab contains more detailed

information on each VLAN network, including the VLAN name, VLAN ID, switches, and the list of ports. At each VLAN network in the View VLAN tab, there is a button that allows for the VLAN topology to be viewed in another window. On the left side GUI, the deleting VLAN function menu allows the network administrator to delete either an individual VLAN or all of the VLANs presented in the network.

## 5.2 Static VLAN

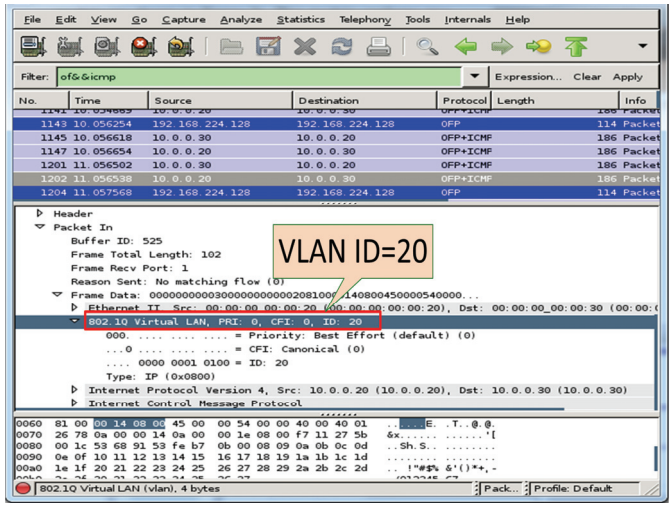


**Fig. 9.** Captured VLAN ID in ping packet between two hosts.

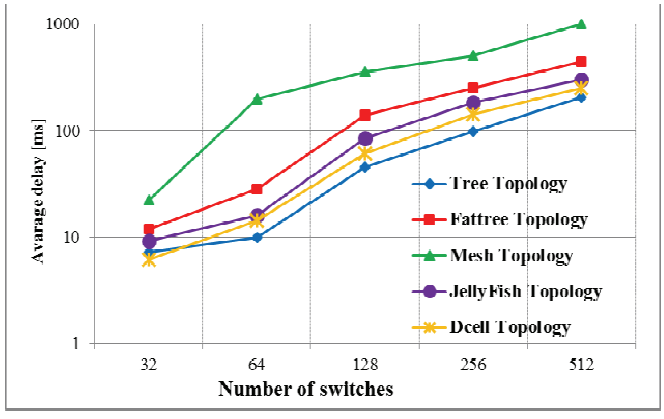


**Fig. 10.** Average delay of dynamic Virtual Local Area Network (VLAN) configuration.

To demonstrate the idea, we set up a real testbed, which is described in Fig. 8. We used four Soekris 5501 devices with OpenvSwitch 1.10 as OpenFlow switches. In this experiment, we wanted to establish two VLAN networks for two pairs of hosts with the VLAN IDs of 10 and 20, respectively. Then, we checked the compliance of the Static VLAN function by inspecting the VLAN ID of two communications between the two pairs of hosts that were mentioned above. Fig. 9 shows the ping packet between two hosts captured by Wireshark. It illustrates that the packet is tagged with a VLAN ID

(VLAN ID = 20) correctly. This means that two hosts can only communicate within this VLAN and that they cannot communicate with other hosts in a different VLAN.

### 5.3 Dynamic VLAN

In this section, we evaluate the average network delay for creating a VLAN segmented network for a pair of hosts in a dynamic manner. We compared this value amongst various types of network topologies built in Mininet [5]. The delay when using dynamic VLAN is calculated as follows:

$$T = T_{pkt-in} + T_{flow\,\mathrm{mod}} + T_{sw} + T_{ctrl} + T_{stp} + T_{query}(ms) \quad (1)$$

Where:

- $T_{pkt\text{-}in}$ and $T_{flowmod}$ are the time for sending a packet-in message from the switch and the time for installing a rule from the controller, respectively.
- $T_{sw}$ and $T_{ctrl}$ are the time for processing the packet at the switch and the time for processing the packet at the controller, respectively.
- $T_{stp}$ and $T_{query}$ are the time for calculating the shortest path tree at the controller and the time for making a query of the VLAN mapping information from the VLAN database, respectively.

We established the experiment based on the Mininet in out-of-band control mode and ran the Floodlight controller in the same Mininet host. This host was running Ubuntu 12.04 LTS with 4G RAM. We observed that the time for sending Packet-in and installing the FlowMod were independent from the type of topology. The latency at the switch and the controller were also quietly consonant. Therefore, T mostly depends on the time for calculating the shortest path tree. Fig. 10 shows the result of T with various types of network topologies and a varying number of switches. We ran several network topologies that are being used for constructing enterprise and campus network architectures, such as Fat-tree [16] and D-Cell [17].

We observed that the procedure of configuration a dynamic VLAN setup using SDN is quite similar to the procedure of traditional dynamic VLAN configuration using VMPS and VLAN Query Protocol (VQP). However, our system is much more flexible and easier because it reduces the complexity of configuring the VTP domain. In addition, it has more scalability because the VLAN database is stored in the controller, which has a global view of the entire network. Moreover, the mobility of the dynamic VLAN host in our approach is more efficient than the traditional one.

## 6. Conclusions

In this paper, we leveraged SDN and OpenFlow into the campus and enterprise network environments. Our main contribution is that we designed and implemented a system for configuring a VLAN network in both a static and dynamic manner. It addresses the complexity of configuring VLANs, which network administrators must deal with when managing VLANs in campus and enterprise networks. In addition, our system provides an interactive graphic user interface for visualizing a VLAN network and offers flexibility of usage to the network administrator. By providing a separated topology view for each VLAN network, the network administrator can easily monitor many

VLAN networks at the same time. Nowadays, VLAN is not only present in campus and enterprise networks, but is also widely used for managing a network in the cloud computing system. We believe that our system can be easily adapted for use in this environment.

In the future, we plan to extend this work to support in a multi-domain, as mentioned in [18]. In addition, the rule placement problem [19] that refers to the correctness of rule installation into switches and the deployment of a new STP algorithm [20] also requires further studies.

## Acknowledgement

This work was partly supported by the ICT R&D program of MSIP/IITP, Republic of Korea. [B0101-15-1351, Distributed & OpenFlow Based Virtual Mobile Core Network]

## References

[1] M. Yu, J. Rexford, X. Sun, S. Rao, and N. Feamster, "A survey of virtual LAN usage in campus networks," *IEEE Communications Magazine,* vol. 49, no. 7, pp. 98-103, 2011.

[2] CISCO System Inc., "Understanding VLAN Trunk Protocol (VTP)," 2014; http://www.cisco.com/c/en/us/support/docs/lan-switching/vtp/10558-21.html?mdfid=280831071.

[3] Open Networking Foundation, "Software-defined networking: the new norm for networks," 2012; https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, 2008.

[5] Y. Jarraya, T. Madi and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1955-1980, 2014.

[6] Floodlight controller [Online]. Available: http://www.projectfloodlight.org/floodlight/.

[7] Mininet [Online]. Available: http://mininet.org/.

[8] S. D. Krothapalli, X. Sun, Y. W. E. Sung, S. A. Yeo, and S. G. Rao, "A toolkit for automating and visualizing VLAN configuration," in *Proceedings of the 2nd ACM Workshop on Assurable and Usable Security Configuration,* Chicago, IL, 2009, pp. 63-70.

[9] S. D. Krothapalli, S. A. Yeo, Y. W. E. Sung, and S. G. Rao, "Virtual MAN: a VLAN management system for enterprise networks," in *Proceedings of ACM SIGCOMM 2009 - demo session,* Barcelona, Spain, 2009.

[10] H. Lu, N. Arora, H. Zhang, C. Lumezanu, J. Rhee, and G. Jiang, "Hybnet: network manager for a hybrid network infrastructure," in *Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference*, Beijing, China, 2013.

[11] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: reaping the benefits of incremental SDN deployment in enterprise networks," in *Proceedings of USENIX Annual Technical Conference,* Philadelphia, PA, 2014, pp. 333-345.

[12] Y. Yamasaki, Y. Miyamoto, J. Yamato, H. Goto, and H. Sone, "Flexible access management system for campus VLAN based on OpenFlow," in *Proceedings of 2011 IEEE/IPSJ 11th International Symposium on Applications and the Internet (SAINT),* Munich, Germany, 2011, pp. 347-351.

[13] A. Lara, A. Kolasani, and B. Ramamurthy, "Simplifying network management using Software Defined Networking and OpenFlow," in *Proceedings of 2012 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS),* Bangalore, India, 2012, pp. 24-29.

[14] P. W. Tsai, P. W. Cheng, C. S. Yang, M. Y. Luo, and J. Chen, "Supporting extensions of VLAN-tagged traffic across OpenFlow networks," in *Proceedings of 2013 2nd GENI Research and Educational Experiment Workshop (GREE),* Salt Lake City, UT, 2013, pp. 61-65.

[15] R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, et al., "Carving research slices out of your production networks with OpenFlow," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 129-130, 2010.

[16] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63-74, 2008.

[17] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75-86, 2008.

[18] V. G. Nguyen and Y. H. Kim, "Virtual application layer for policy management in programmable networks using NFV," in *Proceedings of 5th IEEE International Conference on Communications and Electronics (ICCE)*, Da Nang, Vietnam, 2014, pp. 629-633.

[19] X. N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Optimizing rules placement in OpenFlow networks: trading routing for better efficiency," in *Proceedings of the 3rd Workshop on Hot Topics in Software Defined Networking*, Chicago, IL, 2014, pp. 127-132.

[20] A. G. Furculita, M. V. Ulinic, A. B. Rus, and V. Dobrota, "Implementation issues for modified Dijkstra's and Floyd-Warshall algorithms in OpenFlow," in *Proceedings of 2013 RoEduNet International Conference on Networking in Education and Research*, Iasi, Romania, 2013, pp. 1-6.

**Van-Giang Nguyen**

He received his B.S. degree in electronics and telecommunications from Hanoi University of Science and Technology, Hanoi, Vietnam in 2012, and the M.S. degree in electronic engineering from Soongsil University, Seoul, Korea in 2014. He is currently pursuing his Ph.D. program in Soongsil University. His main research interests are SDN (software defined networking), Openflow, future mobile network architecture, DMM (distributed mobility management), NFV (network function virtualization), SDN and virtualization in mobile network architecture, 5G mobile packet core network, SFC (service function chaining), ICN (information centric networking). He is a member of IEEE SDN and KICS.

**Young-Han Kim** http://orcid.org/0000-0002-1066-4818

He received his B. S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1984, and the M.S. and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology, Seoul, in 1986 and 1990, respectively. From 1987 to 1994, he was a senior member of the research staff in the Digicom Institute of Telemetics, Seoul, Korea, where he did research on data and computer communication systems. Since September 1994, he has been with Soongsil University, where he is a professor of school of electronic engineering. He is a director of Ubiquitous Network Research Center and a director of Convergence-ITRC (Convergence Information Technology Research Center) project supported by MSIP(Ministry of Science, ICT & Future Planning), Korea. His research interests include wireless networking and future networking, SDN (Software Defined Networking), ICN (Information Centric Networking) and sensor networking. He is a member of IEICE and IEEE.