Dynamic Virtual Local Area Network Provisioning in Software-Defined Networking

Mael Spangenberger*, Victor Andreas Boye†, and Jaehoon (Paul) Jeong‡
*Department of Electrical & Computer Engineering, Sungkyunkwan University, Suwon, Republic of Korea
† Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Odense, Denmark
‡Department of Computer Science & Engineering, Sungkyunkwan University, Suwon, Republic of Korea
Email: *maelberger.indp@gmail.com, †viboy20@student.sdu.dk, ‡pauljeong@skku.edu

***Abstract*—In modern networking, the logical segmentation of devices is crucial for improving security, improving network performance, and providing better visibility into device roles and purposes. Virtual Local Area Networks (VLANs) play a key role in achieving this segmentation; however, the manual provisioning and allocation of devices to VLANs can be tedious and error-prone. Software-Defined Networking (SDN) has emerged as a promising solution to address these challenges due to its centralized management, programmability, and automation capabilities. This paper proposes a method for dynamic VLAN provisioning using SDN, leveraging RESTCONF APIs for programmatic control and YANG data models for standardized configuration. The designed system assigns connected devices to appropriate VLANs or creates new VLANs based on predefined criteria such as connection type, physical location, and congestion levels.**

***Index Terms*—VLAN, Dynamic Provisioning, OpenDaylight, YANG, RESTCONF.**

I. Introduction

In computer networking, it is important to be able to logically split up the networking devices, as this can increase security by not allowing access to all network devices from a single connection [1]. It also helps give a better overview of the network devices and what purposes they serve. Furthermore, it can also increase network performance [2]. However, it can be tedious and complicated to provision VLANs and allocate devices to VLANs manually [1], [3]. The advantage of VLANs can be seen in Fig. 1 which shows how the different devices on the network are seperated based on which VLAN they belong to, despite being connected to the same switch.

In more recent years, Software-Defined Networking (SDN) has become more popular because of its advantages in some aspects in network management such as centralized management, automation, and programmability [4], [5]. Because of the advantages of SDN, it is a good candidate to make the VLAN provisioning and allocation be easy for network administrators with the aim of automating the process. To facilitate this goal, RESTCONF [6] can be used to interact with the SDN APIs in order to programatically control the network [7]. Furthermore, for defining the models used to interact with the SDN, YANG data models [8] are used to make sure that the schemes used are correct and interpretable by the SDN Controller.

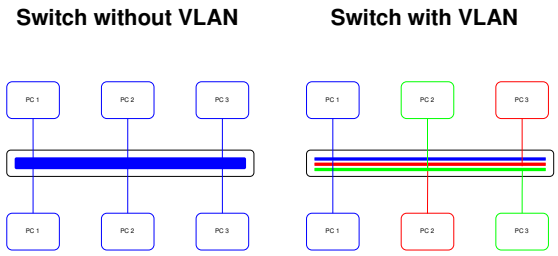

Fig. 1: Showcases the difference between networks with VLANs and without VLANs, different colors represent different VLAN.

The remainder of this paper is organized as follows. Section II summarizes related work. Section III describes the design, and Section IV describes the implementation. Finally, in Section V, the paper is concluded along with future work.

II. Related Work

There have been other attempts at making VLAN management be easy. Krothopalli et al. [9] focuses on VLAN management specifically for traditional enterprise networks. On the other hand, Lu et al. [10] has looked at hybrid networks of SDN and traditional networks and the management of those.

III. Design

Fig. 2 depicts the flowchart of the designed program for VLAN provisioning. The program will query a RESTCONF endpoint of the OpenDaylight controller to check for any newly connected devices. If there are any new devices, the program has to decide which VLAN this device should be allocated to. If there are no suitable VLANs for the device, a new VLAN for it is created, and the device is allocated to that one. If a suitable VLAN is

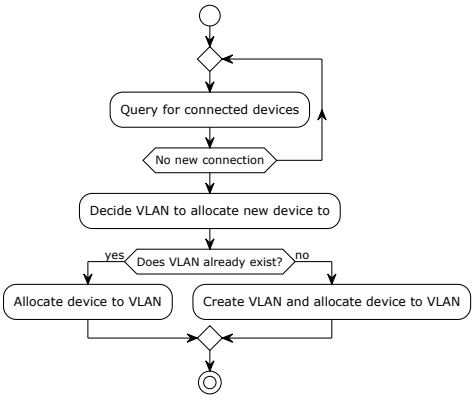

Fig. 2: Flowchart of the VLAN provisioning program.

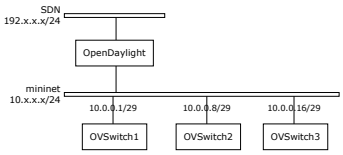

Fig. 3: Example of a VLAN configuration for three switches in Mininet

found, the device can be assigned to the existing VLAN instead.

There can be several different ways to determine which VLAN is suitable for a device. It could be based on the connection to the network, such that wired devices are on a separate VLAN not having wireless devices. In bigger networks, VLANs could be separated by their physical locations. A third way of determining the suitable network could be to have a set size of VLANs. When there are no more spots on a VLAN, a new one is created with the same size as the other VLANs. This could be done in order to limit the amount of network congestion on each VLAN.

In Fig. 3, an example of a network with VLANs can be seen. The example network consists of three switches where each has its own VLAN and a set of IP addresses. In this way, the different devices connected to each switch would be naturally separated. Another way to provision the VLANs could be through a single switch and have multiple VLANs for one switch instead of a VLAN per switch. Fig. 4 shows an example of three VLANs on one switch, which could be a more realistic example in an SDN network.

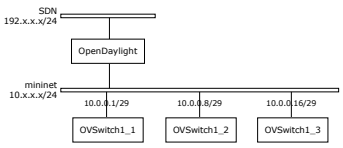

Fig. 4: Example of a VLAN configuration for one switch in Mininet

Now let us look at the VLAN provisioning procedure as follows. First, the VLANs are allocated to an array in the system. This could be the case where there are already VLANs to accommodate hosts in the SDN network. Next, the system gets the current devices available in the network via RESTCONF.

The main loop of the system works as follows. First, the current devices are retrieved from the SDN controller via RESTCONF. These are compared with the already configured devices in the network. If there are more devices currently connected than those in the `Devices` array, the system configures these. It does this by first assigning the new device to a variable for VLAN provisioning, and then deciding which VLAN this device should be assigned to. This function should be implemented based on the criteria with which the system should assign devices to VLANs. As explained earlier, this could be by a location, MAC address, Round Robin, and connection type.

An example where connection type criteria makes sense is an office setting. All office computers could be on a wired VLAN since they are connected via Ethernet. Any "guest" device like a smartphone of either an employee or any guest that comes to the office will be assigned to a VLAN for wireless devices. This setup would increase security by not allowing wireless devices to connect and attack the office computers. All the smartphones logging onto the network will not affect network congestion in the wired VLAN since they are joining the separated, wireless VLAN while all the office computers are on the wired VLAN.

If the VLAN for a certain type of a device does not yet exist, the system should create a VLAN via RESTCONF. After making sure that the VLAN exists, the device gets assigned to the VLAN. Finally, the `Devices` array is updated to make sure that the newly connected device does not get assigned to a VLAN again.

IV. Evaluation

A. Emulation Setup

OpenDaylight is used for the SDN controller and Mininet is used as the emulator for the network. Different network topologies are used to evaluate different scenarios.

B. Results

Unfortunately, we were unable to get the design implemented properly and thus have no real results in an experiment; note that the implementation of our proposed VLAN provisioning method is left as future work. However, we still feel that the design is sound for implementation and could help in VLAN assignment and provisioning.

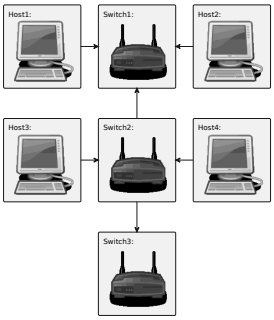

Fig. 5: The topology of an emulation network

```
{
   "flow":{
      "id":"forward-10-4",
      "priority":100,
      "match":{
         "vlan-match":{
            "vlan-id":{
               "vlan-id":10,
               "vlan-id-present":true
            }
         }
      },
      "instructions":{
         "instruction":[
            {
               "order":0,
               "apply-actions":{
                  "action":[
                     {
                        "order":0,
                        "pop-vlan-action":{

                        }
                     },
                     {
                        "order":1,
                        "output-action":{
                           "output-node-connector":
                              "4"
                        }
                     }
         . . .
}
```

Fig. 6: A JSON file for VLAN forwarding configuration

The topology in Fig. 5 is used for testing. However, the Mininet CLI is also used to add a new host during the execution of the VLAN provisioning program, but OpenDaylight does not register the newly connected host with a VLAN because we cannot test the automatic registration capability of the program if RESTCONF never reports the new host. We can check that the program is executed correctly when Mininet first connects to OpenDaylight by launching the program and then launching the Mininet and OpenDaylight containers.

The VLANs are defined using OpenFlow rules in JSON data sent via the RESTCONF API. An example of the JSON data to configure a VLAN forward action can be seen in Fig. 6. This sets up the VLAN forwarding on the switch to port 4 when the traffic matches VLAN 10. The source code is available at GitHub: https://github.com/jaehoonpauljeong/Data-Modeling-Group-4-Project. The demonstration video clip is available at YouTube: https://youtu.be/jx6c-KlLBJs.

V. Conclusion

This paper investigates the challenges of dynamic VLAN provisioning in SDN networks. It introduces the design for a method of dynamically allocating devices to VLANs or provisioning new VLANs. This method can be used to automate and enhance VLAN provisioning based on different parameters, such as congestion, location, and connection. As future work, we will develop other ways of assigning devices to appropriate VLANs and implementing QoS and access lists into the VLANs.

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT (MSIT), South Korea (No. RS-2024-00398199 and RS-2022-II221199). Note that Jaehoon (Paul) Jeong is the corresponding author.

References

[1] P. Garimella, Y.-W. Sung, N. Zhang, and S. Rao, “Characterization Study of VLANs in a Campus Network.”

[2] Y. A. Makeri, G. T. Cirella, F. J. Galas, H. M. Jadah, and A. O. Adeniran, “Network Performance Through Virtual Local Area Network (VLAN) Implementation & Enforcement On Network Security For Enterprise,” *International Journal of Advanced Networking and Applications*, vol. 12, no. 6, pp. 4750–4762, 2021.

[3] R. Chokshi and C. Yu, “Study on VLAN in Wireless Networks,” *Hakupäivä18*, vol. 6, 2007.

[4] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, “Software defined networking: State of the art and research challenges,” *Computer Networks*, vol. 72, pp. 74–98, 2014.

[5] M. Hussain, N. Shah, R. Amin, S. S. Alshamrani, A. Alotaibi, and S. M. Raza, “Software-defined networking: Categories, analysis, and future directions,” *Sensors*, vol. 22, no. 15, p. 5551, 2022.

[6] A. Bierman, M. Bjorklund, and K. Watsen, “RESTCONF Protocol,” *RFC 8040*, Jan. 2017. [Online]. Available: https://datatracker.ietf.org/doc/rfc8040/

[7] B. Ribes Garcia, “OpenDaylight SDN controller platform,” B.S. thesis, Universitat Politècnica de Catalunya, 2015.

[8] M. Bjorklund, “The YANG 1.1 Data Modeling Language,” *RFC 7950*, Aug. 2016. [Online]. Available: https://datatracker.ietf.org/doc/rfc7950/

[9] S. D. Krothapalli, S. A. Yeo, Y.-W. E. Sung, and S. G. Rao, “Virtual man: A VLAN management system for enterprise networks,” *Demo Session, ACM SIGCOMM*, 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*, 2013, pp. 1–6.