2022-07-11

Demonstrating Configuration of Software Defined Networking in Real Wireless Testbeds

Saish Urumkar
Technological University Dublin

Gianluca Fontanesi
University College Dublin

Avishek Nag
University College Dublin, Ireland

See next page for additional authors

Recommended Citation

S. Urumkar, G. Fontanesi, A. Nag and S. Sharma, "Demonstrating Configuration of Software Defined Networking in Real Wireless Testbeds," 2022 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), 2022, pp. 1-2, doi: 10.1109/LANMAN54755.2022.9819994.

Funder: NGIAtlantic H2020 project under agreement no. OC3-292

Authors
Saish Urumkar, Gianluca Fontanesi, Avishek Nag, and Sachin Sharma

Demonstrating Configuration of Software Defined Networking in Real Wireless Testbeds

Saish Urumkar
TU Dublin
Saish.Urumkar@TUDublin.ie

Gianluca Fontanesi
UCD
gianluca.fontanesi@ucd.ie

Avishek Nag
UCD
Avishek.Nag@UCD.ie

Sachin Sharma
TU Dublin
Sachin.Sharma@TUDublin.ie

***Abstract*—Currently, several wireless testbeds are available to test networking solutions including Fed4Fire testbeds such as w-ilab.t and CityLab in the EU, and POWDER and COSMOS in the US. In this demonstration, we use the w-ilab.t testbed to set up a wireless ad-hoc Software-Defined Network (SDN). OpenFlow is used as an SDN protocol and is deployed using a grid wireless ad-hoc topology in w-ilab.t. In this paper, we demonstrate: (1) the configuration of a wireless ad-hoc network based on w-ilab.t and (2) the automatic deployment of OpenFlow in an ad-hoc wireless network where some wireless nodes are not directly connected to the controller. The configuration of OpenFlow is shown using the Fed4Fire GUI (Graphical User Interface). The automatic configuration time of OpenFlow is calculated during the demonstration. Further, data traffic is transmitted from each wireless device to another device to show that an OpenFlow network is set up correctly on the w-ilab.t testbed.**

***Index Terms*—SDN/OpenFlow, Wireless Networks, Fed4Fire**

I. Introduction

Currently, several open testbeds such as Fed4Fire, POWDER, and COSMOS are now being developed to test new protocols and prototypes in real-world contexts. As part of an NGIAtlantic project, [1], we conducted Software-Defined Networking (SDN) and machine learning experiments on the w-iLab.t testbed, which is an emulab-based Fed4Fire testbed [2]. The project utilizes an automatic configuration method for the deployment of SDN in a wireless ad-hoc network proposed in [3]. In this method, OpenFlow is one of the SDN protocols. Previously, this method was tested in a wireless network created using mininet [4] which deploys a virtual OpenFlow network [5].

The purpose of this demo is to demonstrate how SDN/OpenFlow can be configured automatically using the above method in real testbed settings using the jFed tool [6] and the w-iLab.t testbed [2]. According to the topology in use, this configuration requires additional steps such as setting up wireless links. For our demonstration, we consider a grid wireless topology for OpenFlow configuration. The demonstration consists of three parts. In the first part, the topology is configured on the W-ilab.t testbed using the jFed tool, and in the second part, OpenFlow configuration is implemented by using the method proposed in [3]. Lastly, the results are presented. The configuration presented in this demo can be used to deploy OpenFlow in other testbeds, including CityLab, POWDER, and COSMOS.

II. Configuration of OpenFlow in Testbeds

In the Fed4Fire's testbed, we configured OpenFlow using two steps. Below are the steps:

A. Deployment of a wireless topology

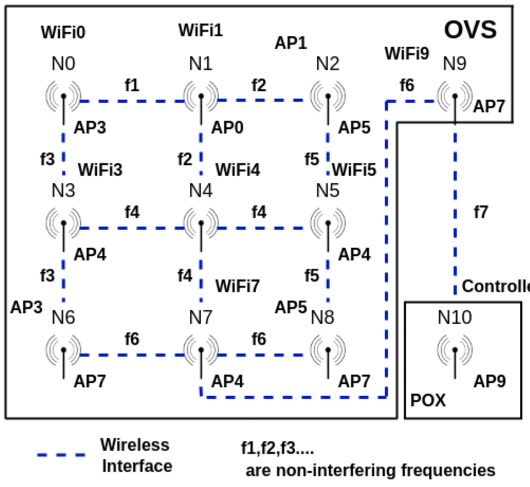

Fig. 1. W-iLab.1 Open Flow Automatic Configuration

With most testbeds, including Fed4Fire wireless testbeds, the nodes are positioned in fixed locations and most of them can be directly reachable to each other. Consequently, creation of a multi-hop topology is not possible on these testbeds without tweaking the parameters of the links. In this demo, we created multi-hop links by using different frequencies ($f1$, $f2$, $f3$ .. etc., as shown in Figure 1) of channels. This scenario is discussed in [1]. In this scenario, access points (AP) with different frequencies, as shown in Figure 1, are used to create different links. For example, in Figure 1, in order to send traffic from node N0 to N2 through node N1, access point AP3 sends traffic to AP0 using frequency f1 and then AP0 sends to AP1 using frequency f2. Further, a unique IP address is assigned to each WiFi interface used in our topology.

We created a grid topology using the above scenario where 10 nodes (nodes N0 to N9) are OpenFlow switches and node N10 is the controller. In this setup, Open vSwitch is used to deploy an OpenFlow switch and POX is used to deploy the controller node.

B. Automatic Configuration of OpenFlow

The problem to configure OpenFlow in the WiFi interfaces was that the links created using the mechanism provided in the previous subsection stop working once they are added to the Open vSwitch. This is because Open vSwitch works based on the IEEE 802.3 MAC protocol, while WiFi links work based on wireless MAC protocols, such as IEEE 802.11. In order to transport traffic from Open vSwitch to WiFi interfaces, we created GRE tunnels, as described in [3]. All the creation of GRE tunnels and addition of the controller information are done automatically using the method provided in [3]. Further, using the above method, the OLSR (Optimized Link State Routing) protocol is used to route the traffic (i.e., control traffic) between the controller and switches and the OpenFlow protocol is used to forward the traffic (i.e., data traffic) between switches. The steps that are executed in our automatic configuration method are below:

1) All nodes (N0 to N10) run OLSR.
2) All nodes except the controller node run Open vSwitch.
3) Node 10 runs the POX controller.
4) Once a switch detects a neighbour using the OLSR, it adds a GRE tunnel to reach the neighbour using the OVS-DB protocol.
5) Once the controller detects a new switch using the OLSR, it adds its own IP address, transport-layer protocol, and port number to the switch using the OVS-DB protocol.

III. Demonstration and Results

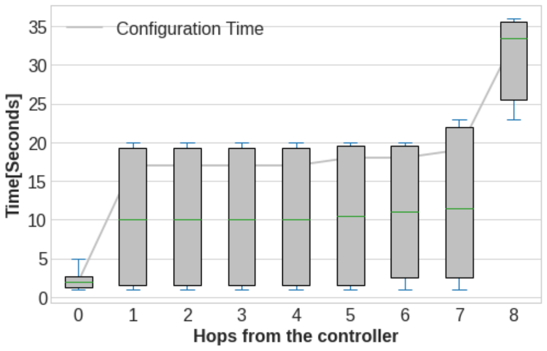

Fig. 2. Automatic Configuration Time

We demonstrate the creation of a wireless grid topology and automatic configuration of OpenFlow using wireless NUC2014 nodes of W-iLab1.t. We configure wireless links and IP addresses by using a multi-command terminal in jFed. Furthermore, we then demonstrate the automatic configuration method of OpenFlow by displaying the OpenFlow sessions established by the POX controller.

During the demonstration, we show that the controller is also able to create sessions with all OpenFlow switches including the nodes that are not directly connected to it (nodes N0 to N8 in Figure 1). The results will be collected for the amount of time taken by the controller to connect to all OpenFlow nodes and to configure the GRE tunnels. The number of hops depends on the path taken by the controller to reach an OpenFlow node. Figure 2 shows the results of configuration time i.e., time taken by the controller node to detect OpenFlow switches, set up the routes, ports, GRE tunnels, and corresponding IP addresses. The results were collected 20 times and box plot was plotted. The nodes located at hop 0 have the lowest automatic configuration time, as these nodes are directly connected with the controller. The automatic configuration time from hop 1 to 7 is approximately the same in our topology. Further, hop 8, has significantly more configuration time because it is the hop that is the farthest and have lowest WiFi channel. These results are different from those reported in [3], where we observed a linear increase in the automatic configuration time with the number of hops increased. This difference is observed, as this setup is run in the testbed and previously, it was run using mininet.

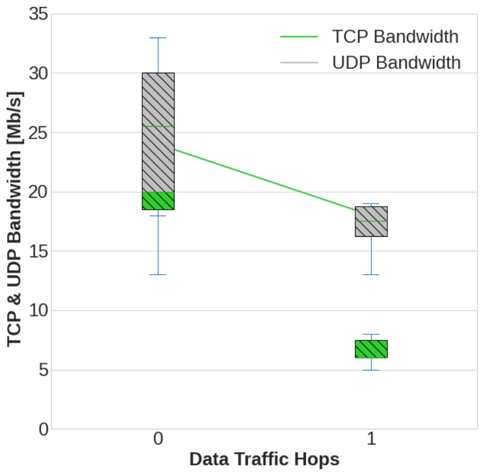

Fig. 3. TCP and UDP Bandwidth Utilization for Data Traffic

We also show the data traffic bandwidth utilization (TCP and UDP) by measuring the bandwidth using iperf. The results from the current setup are shown in Figure 3. The results show that UDP bandwidth is higher than the TCP bandwidth. The demonstration video can be found at [7].

Acknowledgement

This research was funded by the EU H2020 NGIAtlantic project under agreement No. OC3-292.

References

[1] S. Sharma, S. Urumkar, G. Fontanesi, B. Ramamurthy, and A. Nag, “Future wireless networking experiments escaping simulations,” *Future Internet*, vol. 14, no. 4, 2022. [Online]. Available: https://www.mdpi.com/1999-5903/14/4/120
[2] “W-ilab.t.” [Online]. Available: https://doc.ilabt.imec.be/ilabt/wilab
[3] S. Sharma, A. Nag, P. Stynes, and M. Nekovee, “Automatic configuration of openflow in wireless mobile ad hoc networks,” in *HPCS*, 2019, pp. 367–373.
[4] “Mininet.” [Online]. Available: http://mininet.org/
[5] S. Sharma and M. Nekovee, “Demo abstract: A demonstration of automatic configuration of openflow in wireless ad hoc networks,” in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 953–954.
[6] “jfed.” [Online]. Available: https://jfed.ilabt.imec.be/
[7] “Our demo video.” [Online]. Available: https://www.youtube.com/watch?v=kAkrT95tRb4