Demo Abstract: A demonstration of automatic configuration of OpenFlow in wireless ad hoc networks

Sachin Sharma[1] and Maziar Nekovee[2]
[1]National College of Ireland (sachin.sharma@ncirl.ie) and [2]University of Sussex (M.Nekovee@sussex.ac.uk)

***Abstract*—Using OpenFlow, a network can be controlled from one or more servers called controllers. In the demonstration, we show automatic configuration of OpenFlow in a wireless ad hoc network, deployed on a portable testbed, using MININET-WiFi (an emulator for software defined wireless networks). Automatic configuration is shown using a GUI (Graphical User Interface) which shows wireless nodes discovered by the controller. In addition, a video clip is streamed from one node to another and displayed in real time. The demonstration includes automatic configuration in the scenarios in which nodes move from one location to another.**

I. Introduction

A wireless Mobile Ad hoc NETwork (MANET) is an infrastructure less wireless network in which mobile wireless nodes dynamically form a temporary network for communication without the use of a fixed infrastructure. As each node in ad hoc networks takes decisions (e.g., routing) independently, these networks are too complex to manage, too prone to vendor-locking, and too inflexible to adapt to the needs of changing requirements. To overcome these problems there is significant interest from research communities to apply OpenFlow in wireless ad hoc networks [1]. One of the major drivers of OpenFlow is its simplification. It simplifies networks by allowing it to decouple complex software from nodes and deploying it in external servers called controllers.

We propose a method which can automatically configure OpenFlow in a wireless ad hoc network. In this method, we deploy OpenFlow using existing OpenFlow software (Open vSwitch). We consider an ad hoc network (Fig. 1) in which the

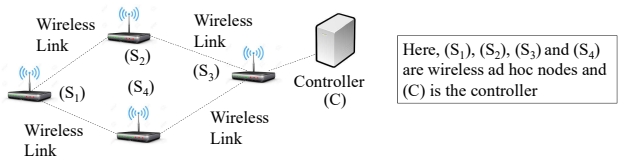

Fig. 1. A Wireless Ad hoc Network

controller is directly reachable to only a few wireless nodes (see node S_3 in Fig. 1). The challenge is therefore that the nodes, which are not directly reachable to the controller, have to find a path to the controller through other nodes (mobile) in the network. This type of an ad hoc network is different from an in-band OpenFlow network, discussed for wired networks in [2]. This is because the nodes in our network can also move from one location to another.

In this demonstration, we show automatic configuration of OpenFlow (using our proposed method) in a wireless ad hoc network, deployed using MININET-WiFi [3]. The demonstration includes automatic configuration in mobility scenarios in which nodes move from one location to another. In addition, video streaming from one node to another, following a path configured by an OpenFlow controller, is demonstrated.

II. Our automatic configuration method

We implement our automatic configuration method in hybrid ad hoc nodes which support both OpenFlow and traditional protocols (such as layer 3 protocols). The followings are two challenges to run OpenFlow in wireless ad hoc networks:

1) Traffic forwarded through an OpenFlow node should be successfully received by a neighbor OpenFlow node over a wireless link.
2) Each wireless node should be able to establish an OpenFlow session with the controller.

The difficulty in overcoming the first challenge is that the MAC addressing scheme of current OpenFlow software (such as Open vSwitch) is based on the IEEE 802.3 standard and wireless ad hoc nodes do not support the IEEE 802.3 standard [4]. They support the ad hoc mode of the IEEE 802.11 standard. To overcome this challenge, we insert tunnels (such as GRETAP) between an OpenFlow node and its neighbor.

The difficulty in overcoming the second challenge is that an OpenFlow session is usually built on top of a transport layer session. Therefore, each node needs an IP address, needs to know the controller IP address and transport layer parameters (e.g., port), and needs to know a path to the controller.

A. Overview of our Method

In our automatic configuration method, communication between the controller and wireless nodes happens on a routing path established by traditional routing protocols and communication between OpenFlow nodes follows a path configured by the controller in the network.

We deploy an OpenFlow controller (shown in Fig. 1) and the OVS-DB server on a wireless node (not shown in Fig. 1) in the network. The controller and OVS-DB server also run the routing protocol (e.g., OLSR). Moreover, all the other wireless nodes run the following protocol stack: (1) an address auto-configuration protocol [5], (2) a traditional routing protocol, (3) a tunnel agent, (4) the OVS-DB client, (5) a transport layer protocol, and (6) the OpenFlow protocol.

Using the address auto-configuration protocol, a wireless node gets an IP address without running the DHCP server in the network. Using traditional routing (e.g., OLSR), a wireless node knows the IP addresses of neighbors and paths to servers

(such as OVS-DB server and the controller). The tunnel agent creates tunnels (e.g., GRE) with all the neighbors discovered by traditional routing. These tunnels are used to transport IEEE 802.3 frames (generated by Open vSwitch) over a wireless link. Using an OVS-DB client, the OVS-DB server configures the controller IP address and OpenFlow related transport layer parameters (e.g., port) in a wireless OpenFlow node. Once all the aforementioned parameters are known, the transport layer protocol establishes a transport layer session between the wireless node and the controller. The path between the wireless node and the controller is decided by the routing protocol. Once the transport layer session is established, the wireless node establishes an OpenFlow session with the controller using the path discovered by the routing protocol.

As wireless nodes may move from one location to another, they may become unreachable from their neighbor nodes. When the path between the controller (or the OVS-DB server) and a wireless node contains an unreachable neighbor, the communication between them does not work until a new valid path (decided by the routing protocol) is established in the network. In case, OpenFlow detects the communication failure, the OpenFlow session is broken and a new session is established, when the routing protocol establishes a new path.

B. Emulations

We have emulated different ad hoc networks - linear, sparse and dense - on the Fed4Fire testbed using Mininet-WiFi. Twenty wireless ad hoc nodes are deployed using MININET-WiFi software. The radio range of nodes is 74 meter. Open vSwitch and the POX controller are used for OpenFlow emulations. OpenFlow detects the communication failure in our experiment in 15 seconds. OLSR is deployed as a routing protocol and OLSR neighbor hold time is kept as 20 seconds. TCP is used as a transport layer protocol between the controller and wireless nodes. The OVS-DB server is located on the controller. All the experiments are run 50 times and minimum, average and maximum values are shown in results.

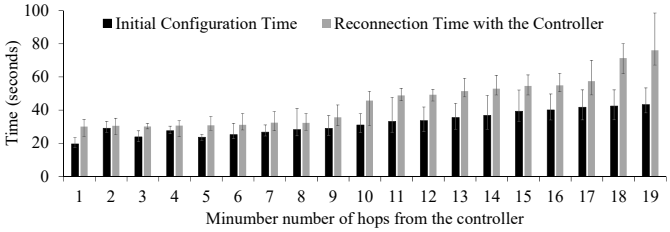

Fig. 2. Automatic configuration time

Fig. 2 shows the results of initial configuration time and the controller re-connection time when nodes move from one location to another. Moreover, we performed data traffic experiments. The results of these experiments show that there is performance degradation in data traffic communication, as we used tunnels to transport the IEEE 802.3 standard frames (generated by Open vSwitch) on wireless links. The performance degradation (in the average delay) was approximately 3.74 ms per packet in our experiments. This performance degradation can be reduced by deploying OpenFlow on high-speed hardware devices.

III. Demonstration

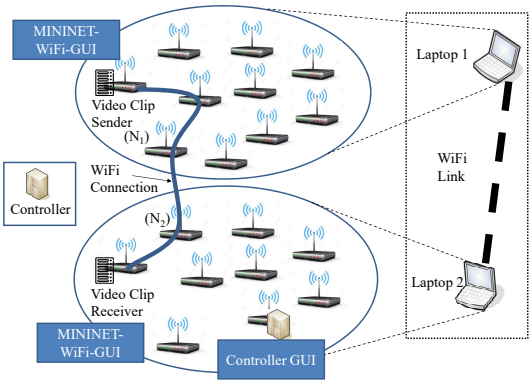

Fig. 3. Demonstration on a Portable Testbed

With the portable testbed (2 laptops connected using a WiFi link, Fig. 3), we demonstrate the automatic configuration of OpenFlow in a MANET, using the similar emulation setting as performed on the Fed4Fire testbed. We emulate 10 wireless ad hoc nodes on laptop 1 and the other 10 wireless nodes on laptop 2, using Mininet-WiFi (Fig. 3). Wireless node N_1 and N_2 (see Fig. 3) are connected using a WiFi link. Therefore, all the wireless nodes of laptop 1 and 2 should be able to communicate with each other using the WiFi link.

The controller and the OVS-DB server are deployed on one of the wireless nodes in the network (Fig. 3). All the nodes run the protocol stack, as explained in previous section. Moreover, for the address automatic configuration protocol, we use the automatic IP addressing scheme, used by Mininet-WiFi [3].

We run our automatic configuration method in the deployed MANET and demonstrate its working using a GUI installed on the controller node. When a wireless node is able to create an OpenFlow session with the controller, the node will be shown in the GUI. In addition to the controller GUI, we show the wireless ad hoc network topology using the GUIs generated by MININET-WiFi on both the laptops.

During the demonstration, we manually move wireless nodes from one location to another using the command line interface of Mininet-WiFi and show in the GUI that the controller has detected it and removed unreachable nodes from the GUI. In addition, when these nodes are able to establish new sessions with the controller, they are again shown in the GUI. Moreover, we stream a video clip from one of the nodes in laptop 1 to another node in laptop 2. We will see that the controller is able insert forwarding entries (using Open vSwitch commands) in the network to send video traffic. The video client on laptop 2 is then able to display it in real time.

References

[1] P. Bellavista et. al., MANET-oriented SDN: Motivations, Challenges, and a Solution Prototype, WoWMoM, 14–22, 2018.

[2] S. Sharma, et. al., In-band control, queuing, and failure recovery functionalities for openflow, IEEE Network, vol. 30(1), 106-112, 2016.

[3] Mininet-WiFi: https://github.com/intrig-unicamp/mininet-wifi

[4] M. Rademacher et. al., Experiments with OpenFlow and IEEE802.11 Point-to-Point Links in a WMN, ICWMC, 2016

[5] A. Munjal, Address Auto-Configuration Protocols and their message complexity in Mobile Adhoc Networks, PhD Dissertation, IIT, 2015