SDN-based MANETs Using Existing OpenFlow Protocol

Saleh Rabia
School of Mathematical and Computer sciences
Heriot-Watt University
Edinburgh, Scotland UK
email res5@hw.ac.uk

Skloul Idris I
School of Mathematical and Computer sciences
Heriot-Watt University
Edinburgh, Scotland UK
email i.s.ibrahim@hw.ac.uk

Georgieva Lilia
School of Mathematical and Computer sciences
Heriot-Watt University
Edinburgh, Scotland UK
email L. Georgieva @hw.ac.uk

***Abstract*—There is a continuous business need for network technologies to increase in functionality, performance and complexity. However, the present network paradigms show a lack of adaptability and are limited to single domain management. Thus, management of the network places a burden on the network's users. In addition, the high number and variety of stationary or dynamic devices make the network massive and intractable, with a complexity that leads to scalability challenges. Modern requirements cannot be supported by the current decentralized mobile ad hoc networks (MANETs) standard models. Additionally, MANETs suffer from packets/network overheads due to topology changes with the distributed and (decentralized) routing in each node. In a typical architecture, the mobile node is responsible for dynamically detecting other nodes in the network. The node can communicate directly or via an intermediate node, and to specify a route to other nodes. Thus, the node takes a decision with only a limited view of the network's topology. To this end, the deployment of the Software Defined Networking (SDN) paradigm has the potential to enable the redesign of these models. SDN provides a global view of network topology and a programmable network with centralized management. In this paper, we propose a new architecture for SDN-based MANETs, which is adding an Open Virtual Switch (OVS) per node to find the effect of OVS on the MANETs performance. We present a practical implementation for the new architecture using existing OpenFlow protocol. The tests have been carried out in an emulation environment based on Linux Containers (LXC V 2.0.11), Open Network Operating System (ONOS V 2.5.0) as a remote controller, NS3 and Open virtual Switch (OVS V 2.5).**

Keywords*—*MANET; Software Defined Networking (SDN); Linux Containers (LXC); NS-3; OpenFlow protocol.

I. INTRODUCTION

A collection of mobile nodes able to communicate without any need for fixed infrastructure or administration is called a MANETs. The MANET's nodes have a limited wireless transmission range and energy. In traditional MANETs, MANETs cooperative mobile nodes take decisions (e.g., routing) independently based on their limited view and without global network topology knowledge. Routing protocols run in each node in a distributed way. In addition, the routing decisions which are taken by the mobile nodes are very difficult and sometimes need all mobile nodes to participate in this process, which may lead to high energy consumption and overheads [1]. Furthermore, these networks are prone to vendor-locking, are not flexible enough to allow them to be updated considering the modern requirements of a system, and are complex to manage [2]. According to recent estimates, SDN is expected to grow from $13.7 billion in 2020 to $32.7 billion by 2025, which represents an annual growth rate of 19% [3]. Thus, the SDN model is considered a new generation of networking. The SDN paradigm has shown its effectiveness in wired networks, such as data centres. However, deployment of SDN in MANETs has attracted very little attention and remains an open research problem. The architectures of the completely decentralized MANETs are considered as one of the main reasons they are not used for large topologies. With the SDN paradigm, this philosophy can be revisited. SDN provides centralized control, and a programmable and wide view network [4]. However, due to the dynamic environment of the MANETs, it is challenging to deploy SDN within it.

The Open Networking Foundation (ONF) [5] which is the organization responsible for the OpenFlow protocol defines SDN as "decoupling the data plane from control plane where the networking devices are controlled or updated using the open flow protocol and centralized SDN controller". SDN controller is an application (software) that works as a strategic control point to mange the whole network. Recently, SDN has shed new light on how to control and manage mobile adhoc networks. The control and management of a MANETs with an SDN paradigm is offered more flexibility and enable new features and services in the network. Moreover, the SDN paradigm provides a centralized network where all control plane and management plane functions are pushed to a centralized unit known as the SDN controller. In other words, SDN allows network operators/managers to define policies and/or behaviours of the network in the central controller by making control decisions for each new flow. The SDN controller is effectively the brain of the network, and defines the collaborative behaviour of the network and directs the participating nodes on how to behave. The aim of this work is to take advantage of the efficiency of SDN and implement this paradigm in MANETs using an existing open flow protocol. Additionally, each node is configured to support open flow, which will logically decouple the data plane and control plane. In other words, each node needs to have an OpenFlow switch and acts as router and end host at the same time. Figure 1 considers MANETs where the ONOS controller [6] is directly reachable for each mobile node (out-of-band mode).

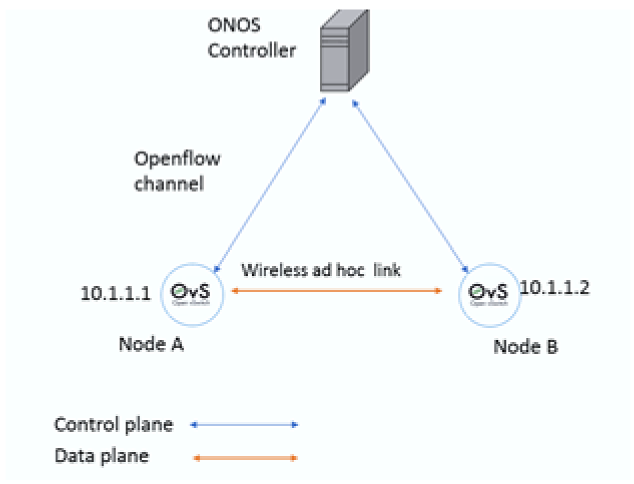

Figure. 1: Network topology.

The most challenging issue facing the SDN-based MANETs architecture is the dynamicity of the network topology, where MANETs nodes (e.g., vehicles and drones) move with a different direction and speed, in addition to security and reliability in the control plane to data plane communication between the SDN controller and OpenFlow-enabled nodes. There are two main approaches that have been proposed to address this problem, classified as out-of-band and in-band modes. Depending on the network architecture, the connection between the controller and forwarding devices can be either out-of-band, in which a different network is used for control communication, or in-band, in which the same network is used for both control communications and data [7]. The out-of-band mode is widely used because of its reliability, as it completely isolates control traffic from data traffic. However, it suffers from scalability issues, and thus, out-of-band mode is used in this paper.

This paper is organized as follow. Section II gives a brief background of OpenFlow based SDN. Section III discusses related work. The proposed SDN-based MANETs architecture is described in Section IV, and the emulation scenario is presented in Section V. Additionally, in the same section, some emulation results are shown and discussed. Finally, Section VI concludes the paper.

II. OpenFlow based SDN background

The most important feature of OpenFlow based SDN is the separation of the control plane and data plane. In addition to this, the SDN paradigm provides programmable, simplified network configuration and management by pushing all control tasks to a centralized controller. An OpenFlow based SDN network is created by OpenFlow switches that communicate with one or more controllers and forward data packets using OpenFlow protocol.

A controller creates and configures the forwarding behaviour by setting rules in flow tables and sending them to each switch in the network. The rules consist of match criteria and actions. There is one or more flow tables in each OpenFlow switch. Thus, it is important to note that matching starts at the first flow table and may continue to additional flow tables in the pipeline [8]. When a packet arrives at the OpenFlow switch, it will first start in flow table (0) and check those entries based on priority. The packet with the highest priority will match first. if the flow needs to continue to another table, a “goto” statement tells the packet to go to the table specified in the instructions. If there is no match found in a flow table, the outcome depends on configuration of the table-miss flow entry. The actions define the packet processing to be applied by the switch, such as packet modification, output forwarding port or forwarding to the controller or flooding.

The OpenFlow protocol is used to enable the controller to install, update and delete the flow entries in the flow table of a switch in a reactive or proactive way. In the reactive form, this is in response to incoming packets, and the OpenFlow switch cannot make forwarding decisions, functioning as a dumb switch, and if the incoming packet does not match any rules in the flow table, the switch sends the packet to the controller as packet-in messages. The controller may perform different actions depending on its configuration. For example, the controller may calculate the path for this packet and send update flow entries to the requesting switch as packet-out messages.

An OpenFlow switch can operate at layer 2, layer 3 and up to the transport layer, according to the OpenFlow specification [8]. Thus, it can implement match criteria and actions at different protocol layers. The flexibility of the multi-protocol makes it possible to customize an OpenFlow switch to realize traditional network devices, such as an IP router, Ethernet, firewall, etc. In the OpenFlow switch/SDN, the control plane and data plane are separated. The control plane is processed by software called a “controller”, in which decisions are made, but the data plane is still implemented inside the switch. This is very different from the traditional router/switch, where routing is distributed in each device. In the legacy router/switch, IP processing occurs when a packet arrives at the router, and the routing table associates the IP address of the destination with the next hop address and an outgoing interface. For instance, with an Ethernet as an interface, the router resolves the IP address of the next hop into a MAC address and forwards the packet by rewriting the MAC addresses. Furthermore, the MAC address of the destination becomes the resolved next hop MAC address and the MAC address of the source becomes the router interface MAC address. This is not the case in the Open virtual Switch (OVS)/SDN, where IP routing is replaced by the flow table. In an OpenFlow switch, a match field is used to match packets, consisting of the IP address source/destination, input port, MAC address source/destination etc.

A. *SDN provides flowing functionality*

SDN simplifies network operations, where it reduces complexity by decoupling the control and data planes, while making automation highly secure and scalable.

- It builds programmable networks.
- It has Easy Management (managed remotely), wherein SDN enables the administrator to manage their entire network as a single unit.
- It eliminates manual configuration. In legacy networks, the network is configured manually. Manual processes are usually resource-inefficient, cumbersome, complex and may be error-prone. With the SDN model, a network administrator/ operator can configure all the forwarding devices from a single unit (controller) using applications/API, and can test the configuration before pushing it to all network devices.
- Applications and services are deployed faster by leveraging open APIs.
- SDN provides the possibility of configuring and managing the network using software rather than hardware. It enables network administrators to configure their networks using a software application, rather than changing the configuration of physical hardware.
- There is a centralized MANETs and global view of the network topology. In open SDN, the brain is removed from the networking devices/nodes and placed into a centralized controller. Thus, SDN provides a holistic view of the network.
- Web Graphical User Interface (GUI) centralized controller: the Open Network Operating System (ONOS) controller web GUI is a single page web application which provides a visual interface for the controller [9].

III. Related work

With the rapid development in wireless devices and the increased amount of data traffic transferred through channels, future expectations are that increase in wireless traffic will be larger than for wired traffic.

Few recent works have attempted to apply an SDN model in wireless networks and MANETs to improve performance. Kadhim, Seno and Shihab in [1] have proposed a new routing protocol called the SDN-Cluster Based Routing Protocol (S-CBRP). In the proposed architecture, each cluster head works as a local SDN controller which relies on implementing an SDN agent in each node to manage one or more clusters. These local controllers connect to the central controller, which manages the whole network. Their proposed SDN architecture and routing protocol can increase the lifetime of the network and reduce delay in the route building/rebuilding process.

In [2], the authors provide an innovative method that can automatically configure the open flow in MANETs. The proposed method can be implemented in hybrid OpenFlow switches which support both OpenFlow and traditional routing protocols. The method was tested on MANETs that consisted of 20 nodes in less than 40 seconds, mobility was considered in each node, including the POX controller [10]. The results of the data traffic experiments show that the performance of the network using OpenFlow is lower than for a network that does not deploy OpenFlow. This is due to the use of tunnelling, such as in Generic Routing Encapsulation (GRE) and Virtual Extensible LAN VXLAN between the network nodes.

SDN integrated with wireless mesh networks (WMN) is proposed in [11], in which Detti et al. propose a hybrid approach in the OLSR-to-OpenFlow (O2O) architecture. The OLSR protocol is used in emergency conditions, such as the controller failing or being unreachable, in which the OLSR pushes rules in the OpenFlow switch. Additionally, OLSR is used to send updated network information to the POX controller if the network topology changes. The main disadvantage in this work is related to network updates, because updates to the network require extensive collection of information from the network when there are topology changes, which makes the updates extremely slow. The MANETs SDN-based quality management architecture proposed in [12] provides high flexibility by deployment of new flow management rules at provisioning time and ability to properly handle nodes join/leave event to reduce the network overhead. Authors in [13] have attempted to improve the route finding process. Each node has two different channels, the first is for the communication between the MANET's nodes and the second channel is used for connectivity between the MANET-Controller (MC) and mobile nodes. MC has a global view of the entire topology and receives updates of topology changes from the nodes. The results showed that including the bandwidth in the route finding process increases the reliability of transmissions and the network performance. However, delays slightly increase due to collisions in case where two nodes start transmitting at the same time.

An architecture for an SDN based MANETs is presented in [14]: however, the authors consider one hop links for control communication between mobile nodes and a controller. Additionally, an SDN bridge is used to transfer the OpenFlow frame over wireless links by making modifications on the Reactive Forwarding app. Our approach is almost the same, but we have not made any modifications to ONOS applications. In our proposed approach, each mobile node has a pure OVS (as one device) and works as forwarding and end device as well. Furthermore, each node connects directly to a remote controller (out-of-band mode). The purpose of our contribution is to implement a new centralized MANETs architecture by using SDN paradigm and study their effect on the entire network performance. Thus, to evaluate the impacts of using existing OpenFlow protocol which was originally designed for wired networks on MANETs performance and management.

IV. THE PROPOSED SDN-BASED MANETS ARCHITECTURE

This section provides implementation details through which to understand the proposed SDN-based MANETs architecture.

The connection between nodes is in ad-hoc mode, which is implemented on the node. If OVSs have wireless ports and connect wirelessly, then a 4addr mode/WDS (wireless distribution system) or tunnels, such as VXLAN or GRE as proposed in [2] must be used. An alternative approach described in [14] uses an SDN bridge. In [15], the authors extensively study the use of 4addr mode in OpenFlow protocol/SDN. These options are used to resolve the problem of transfer of OpenFlow packets over a wireless link. OVS is based on the standard specification for Ethernet IEEE 802.3. However, the MANETs do not support the IEEE 802.3, and thus, this frame cannot be successfully transmitted or received over a wireless link unless it is retransformed to an IEEE 802.11 frame.

This work uses an emulated environment built with NS-3, Linux containers (LXC) [16] and an external ONOS controller [6]. An LXC is a set of processes that are detached from the rest of the Operating System (OS). These containers must be compatible with the underlying OS because LXCs share the kernel: only Linux distros (packages) can run with LXC (e.g., Fedora, Ubuntu, Gentoo, Debian. etc.), while Windows or any other OS cannot be run with LXC. Traditional virtualization, such as VMWARE and Virtual Box needs a full OS image for each instance. Thus, any OS can be run using traditional virtualization.

Ubuntu is selected as underlying OS and LXC. OpenFlow virtual Switches (OVS) [17] are attached to the LXCs. Each NS-3 node has a switch and each OVS has two ports: one connects to the controller, while the other connects to the NS-3 node using TapBridge in NS3 and Linux TapDevice. The TapBridge Model [18] allows the replacement of a particular NS-3 simulation node with real hosts. This module overwrites the MAC address of the NS-3 device (node) with the overlying real host MAC address. The real host considers the NS-3 net-device as a local device and all the NS-3 node's incoming traffic will be sent by TapBridge through a virtual TAP interface which is connected to the LXC that host OVS through Linux Bridge, as Figure 2 shows. Moreover, the TapBridge model sends all outgoing traffic through the emulated wireless Ad-Hoc network. Thus, using the underlying network created by NS3, real devices can communicate with each other. The bridge and tap are configured completely outside NS3. Thus, TapBridge in NS3 uses an existing TAP interface previously configured and created by the user [18] [16] [19].

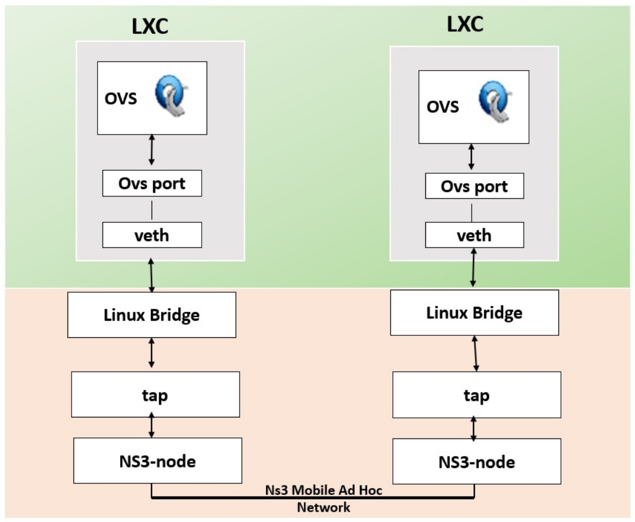

Figure. 2: Internal architecture of mobile Nodes.

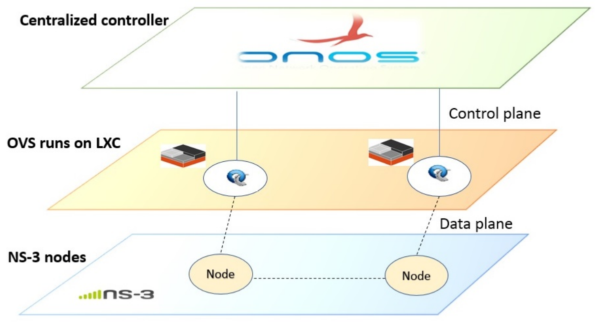

Figure. 3: Global view of the network.

The network is connected as out-of-band mode, as Figure 3 shown, the NS3 nodes are connected wirelessly using ad-hoc network mode. The first layer is remote ONOS controller, the second layer is OVSs where control plane messages are exchanged between the controller and OVSs. while the third layer consists of NS-3 nodes, which includes data plane traffic.

One of the main strengths of this work is that the system works with real applications, e.g., real hardware, Linux Containers, Virtual Machines, that can be directly used in real infrastructure afterwards. The framework is suitable for emulating both wireless (infrastructure and ad hoc) and wired networks. Because this implementation is designed to deploy MANETs scenarios, all the network scenarios provided are based on 208.11 Wi-Fi technology.

V. EMULATION SCENARIO

Our first approach was an attempt to adapt the OpenFlow switch13 module in NS3, as shown in Figure 4 to work with the MANETs by making some modifications in the NS3 mod-

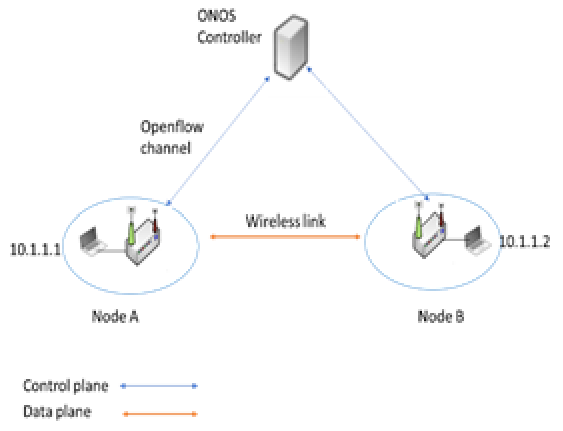

Figure. 4: First model (approach).

ule (source code), such as WifiNetDevice, Ad-hocWifiMac etc. We added Wi-Fi ports to OpenFlow switch, and each switch has two interfaces on the Wi-Fi port: one to connect the switches in ad-hoc mode; and the other one as an Ethernet port to connect the NS-3 node to the Wi-Fi switch. WDS technology [20] has been used in Wi-Fi ports to successfully transmit or receive OpenFlow packets over wireless. However, the communication was unsuccessful, due to NS-3's limited ability to enable OpenFlow.

The second approach was an adaption of an emulation environment using NS-3, LXC and OVS, as shown in Figure 1, where successful connection and data transfer was achieved. The system contains Virtual Box as a virtualization environment, on which runs Ubuntu 16.04 LTS image as an operating system (virtual machine VM), with 29 GB, 4096 MB RAM and 2 CPUs. NS-3, an LXC container attached to OVS, OpenFlow protocol and an ONOS controller running locally in a single machine with NS-3 and LXC. The experimental scenario consists of two mobile nodes: each node has OVS that are connected directly to an ONOS remote controller in out-of-band mode. Each node acts as forwarding and end device. The OVS consists of pure OpenFlow switches (fail mode set as secure). The difference between the approaches is that in the first approach, Wi-Fi interfaces are added to the switches and WDS enabled in the interfaces, while in the second approach, Wi-Fi is already implemented in the NS-3 nodes. In addition, the OVS and NS-3 nodes work as one device. Table 1 shows the software used in the model's implementation and experiments. Table 2 illustrates the network parameters.

To evaluate performance based on the new approach, we characterize three relevant metrics. These metrics are throughput, packet loss, and whether the controller is up or down. The performance measurements were carried out with traffic generated using the well-known tool iperf [21] in the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) for determining the performance of TCP and UDP flows over an SDN-based MANETs. In addition, the Wireshark tool was used to capture packets on interfaces of the network. Each test was executed more than ten times, while the simulation

TABLE I: SOFTWARE USED IN IMPLEMENTATION

Software	***Function***	***Version***
Ubuntu	VM OS	16.04.6
NS3	Nodes, mobility and 802.11 Link Emulation	NS-3.29
OVS	Forwarding devices (data plane)	2.5.0
Iperf	Throughput Measurement	V2
LXC	Ubuntu / guest (virtualized) operating system	2.0.11
ONOS	controller Control plane	2.5.0
OpenFlow	Protocol	1.3

TABLE II: NETWORK PARAMETERS

Parameter	***Value***
Propagation loss	Friis model
Network size	2 nodes, 2 OVS switches
Mobility	Random waypoint
Node speed	5m/s
Simulation area	100 x 200 m2
Simulation runtime	100s
Packet type	TCP / UDP
Traffic Size	1470 bps
TCP widow size	128 KByte
UDP buffer size	208 Kbyte
Wi-Fi standard	802.11a
Link	Ad hoc mode

run time was 100 seconds. These metrics may help the mission planner to gain an insight into the SDN-based MANETs using OpenFlow protocol. Additionally, these metrics show the effect of using a centralized unit (controller) and OpenFlow protocol in MANETs. TCP and UDP data traffic were generated and transmitted, from node B to node A, with node B as server and node A as client as shown in Figure 1, sending 1470 bits per second. Figure 5, and 6 show the TCP and UDP data traffic between two mobile nodes with the parameters stated in Table 2. When node A goes out of range of node B the connection is lost from about 23 seconds to 75 seconds. When after that, node A comes into the communication range of node B, it takes some time (delay) to start the data transmission again because the OVS sends packet-in to the controller requesting new flow table entries, and the controller sends packet-out to the OVS to install new entries. Thus, this process takes some seconds. The number of packets (TCP and UDP) that are lost in the network is shown in Figure 7.

These losses are due to the connection being lost when node leaves and joins the network. As UDP is a connectionless protocol, it is more likely that packets may be lost or that packets arrive out of order at their destination through transmission, whereas TCP is a connection-oriented and reliable

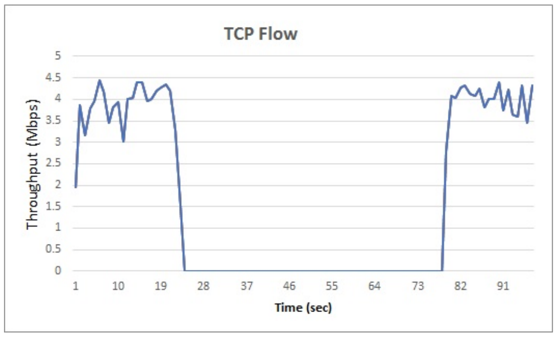

Figure. 5: Throughput in TCP flow.

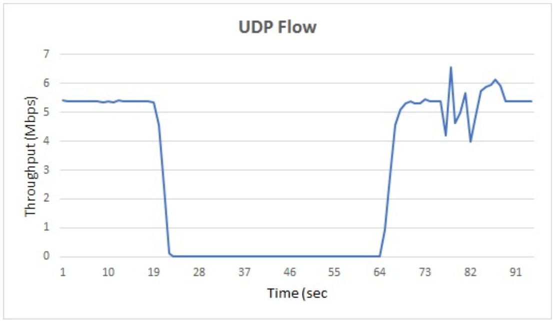

Figure. 6: Throughput in UDP flow.

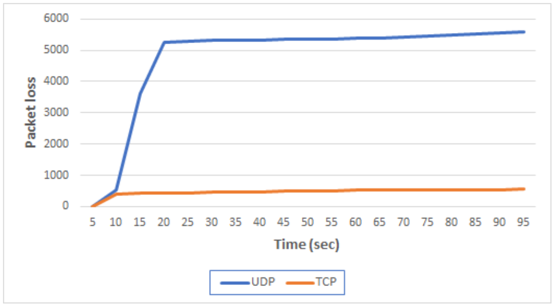

Figure. 7: Packet loss.

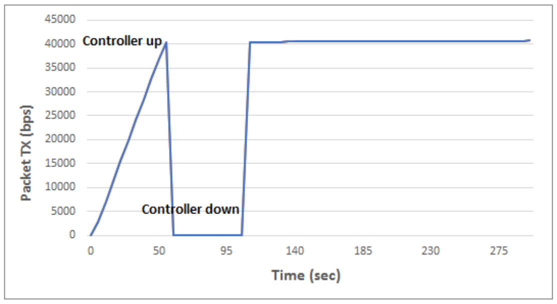

Figure. 8: Controller up/down.

data transfer protocol. Thus, as Figure 7 shows, the packet loss in UDP traffic is much higher than in TCP.

Pure OVS was used for this study, in which the switch does not support traditional route/switch protocols.Figure 8 illustrates when the controller is up or down. Therefore, when the controller is down, there is no connection between the network nodes. On the other hand, the switches do not have a flow table installed on them (being dumb switches), and thus, they cannot forward packets by themselves without the controller.

VI. Conclusion and Future Work

The SDN paradigm has been increasingly extending its presence outside wired networks (e.g., datacentres, intranets) to wireless dynamic environments, such as MANETs.

In this work, a new proposed network architecture was described and discussed which showed how SDN may be applied in MANETs. Additionally, the paper presented a practical implementation of a centralized SDN based MANETs using existing OpenFlow protocol. The centralised control, and network applications which are provided by the controller make the network programmable and easy to manage. Pure OpenFlow switches were used in the implementation, which do not support traditional routing. Mobility is considered in terms of nodes only, but not including the controller. The results of our proposed approach are showing an average of 4 Mbps throughput for the TCP traffic and more than 4.5 Mbps in case of UDP, which is reasonable throughput. The reason for increased packet loss is due to nodes joining /leaving the network. The benefits of the SDN paradigm can be leveraged using a central remote controller and OVSs/OpenFlow enabled switches in the challenging environment of MANETs. The limitations of our proposed approach are due to the one hope link between each mobile node and the controller, and the single point controller. In both cases, there will be no communication at all in case of a controller failure, or no communication with a specific node in case of its single link failure. These two main limitations will be addressed in the future work as well.

In our future work, we plan to create larger complex scenarios and use multiple controllers in the network to avoid a single point of failure in a centralized system, which will also overcome scalability and congestion issues. Additionally, we will use the latest version of OVS (e.g., 2.15), and will compared our approach to the existing solutions which are provided in the related work.

References

[1] A. Kadhim, S. A. Hosseini Seno, and R. A. Shihab, “Routing-protocol for sdn-cluster based manet,”Journal of Theoreticaland Applied Information Technology, vol. 96, pp. 5398–5412,2018.

[2] S. Sharma and M. Nekovee, “Demo abstract: A demonstration of automatic configuration of openflow in wireless adhoc networks,” inIEEE INFOCOM 2019-IEEE Conference onComputer Communications Workshops (INFOCOM WKSHPS),pp. 953–954, IEEE, 2019.

[3] “Software defined networking market size, share and global market forecast to 2025,” 2020. accessed Apr. 09, 2021.

[4] K. Poularakis, G. Iosifidis, and L. Tassiulas, “Sdn-enabled tactical ad hoc networks: Extending programmable control to the edge,” IEEE Communications Magazine, vol. 56, no. 7, pp. 132–138, 2018.

[5] ONF, “Software-defined networking (sdn) definition.” accessed 1. 05, 2021.

[6] “Open network operating system (onos).” accessed Mar. 30, 2021.

[7] A. Dusia, V. K. Mishra, and A. S. Sethi, “Control communication in sdn-based dynamic multi-hop wireless infrastructure-less networks,” in 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp. 1–6, IEEE, 2018.

[8] “Openflow switch specification 1.3.0,” 2013.

[9] A. Koshibe, “The onos web gui,” 2020. accessed Mar. 10, 2021.

[10] “noxrepo/pox: The pox network software platform..” accessed Apr. 03, 2021.

[11] A. Detti, C. Pisa, S. Salsano, and N. Blefari-Melazzi, “Wireless mesh software defined networks (wmsdn),” in 2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob), pp. 89–95, IEEE, 2013.

[12] P. Bellavista, A. Dolci, and C. Giannelli, “Manet-oriented sdn: Motivations, challenges, and a solution prototype,” in 2018 IEEE 19th International Symposium on ”A World of Wireless, Mobile and Multimedia Networks” (WoWMoM), pp. 14–22, 2018.

[13] K. Streit, N. Rodday, F. Steuber, C. Schmitt, and G. D. Rodosek, “Wireless sdn for highly utilized manets,” in 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 226–234, IEEE, 2019.

[14] C. Y. Hans, G. Quer, and R. R. Rao, “Wireless sdn mobile ad hoc network: From theory to practice,” in 2017 IEEE International Conference on Communications (ICC), pp. 1–7, IEEE, 2017.

[15] M. Rademacher, F. Siebertz, M. Schlebusch, and K. Jonas, “Experiments with openflow and ieee802. 11 point-to-point links in a wmn,” ICWMC 2016, pp. 100–105, 2016.

[16] “How to use linux containers to set up virtual networks nsnam.” accessed May. 01, 2021.

[17] “Open vswitch.” accessed Mar. 20, 2021.

[18] “Ns-3: Tap bridge model.” accessed May. 02, 2021.

[19] V. Sanchez-Aguero, F. Valera, B. Nogales, L. F. Gonzalez, and I. Vidal, “Venue: Virtualized environment for multi-uav network emulation,” IEEE Access, vol. 7, pp. 154659–154671, 2019.

[20] “ns-3-patch-wifi-wds.diff ·.” accessed Apr. 28, 2021.

[21] “iperf - the tcp, udp and sctp network bandwidth measurement tool.” accessed Mar. 23, 2021.