Wireless SDN Mobile Ad Hoc Network: from Theory to Practice

Hans C. Yu, **Giorgio Quer**, and **Ramesh R. Rao**
University of California, San Diego – La Jolla, CA 92093, USA

***Abstract*—A promising approach for dealing with the increasing demand of data traffic is the use of device-to-device (D2D) technologies, in particular when the destination can be reached directly, or though few retransmissions by peer devices. Thus, the cellular network can offload local traffic that is transmitted by an ad hoc network, e.g., a mobile ad hoc network (MANET), or a vehicular ad hoc network (VANET). The cellular base station can help coordinate all the devices in the ad hoc network by reusing the software tools developed for software-defined networks (SDNs), which divide the control and the data messages, transmitted in two separate interfaces. In this paper, we present a practical implementation of an SDN MANET, describe in detail the software components that we adopted, and provide a repository for all the new components that we developed. This work can be a starting point for the wireless networking community to design new testbeds with SDN capabilities that can have the advantages of D2D data transmissions and the flexibility of a centralized network management. In order to prove the feasibility of such a network, we also showcase the performance of the proposed network implemented in real devices, as compared to a distributed ad hoc network.**

I. Introduction

The amount of data traffic transmitted through wireless networks is constantly increasing, and soon it will be larger than the amount of wired traffic, according to a recent report from Cisco [1]. In the future 5G scenario, the pattern of wireless traffic will generate an increasing demand for local traffic, i.e., data connections where the source and the destination of the traffic flow are in close proximity. Examples of devices producing this type of traffic include Internet of Things (IoT) sensors in a smart home or a smart city [2], smartphones of friends that are sharing a video clip in a crowded arena [3], or autonomous driving vehicles that need to share local real-time information about traffic or the presence of obstacles on the streets.

Using current network standards, even local data is transmitted through the cellular network to a centralized entity and then redistributed. This approach is clearly suboptimal for the exchange of local traffic. A more promising approach relies on device-to-device (D2D) technologies [4], where the devices can be organized in a distributed way in a mobile ad hoc network (MANET) [5], or in a vehicular ad-hoc network (VANET) [6] in the case of connected vehicles. A practical implementation of such a network, including some optimization capabilities, can be found in [7].

This type of network is characterized by the absence of a central entity. The central entity, called access point (AP) in a Wi-Fi network or base station (BS) in a cellular network, is the center of the start topology, which can organize the network and route every data packet to the intended destination. Without centralized control, each node has to act independently, make routing decisions and, in the case of a mobile network, dynamically adapt to the fast changing topology. In the literature, several solutions have been proposed to make routing decisions in a distributed way, and also to cope with a dynamic network. In particular, in cases where the traffic is sporadic, the solution is a reactive protocol finding a routing path only when needed, like the on-demand distance vector (AODV) protocol [8]. In the case of a more regular traffic between several source-destination pairs, a proactive protocol defining *a priori* the route between each possible pair of nodes in the network may work better, e.g., the optimized link state routing (OLSR) protocol [9].

The main problem with the latter approach is that it requires frequent control message exchanges between nodes to maintain an updated topology of the network at each node. These control messages may cause a high overhead and, more importantly, in the case of a fast changing topology, they may not be sufficient to provide the updated topology information at each node.

In order to reduce the number of control messages, a hybrid approach between proactive and reactive routing protocols is proposed in non-topology-based routing. According to this approach, the network is divided into clusters and, if source and destination nodes are within the same cluster, a proactive routing strategy like OLSR is used; otherwise, a reactive strategy such as AODV is adopted. An algorithm of this kind is the zone routing protocol (ZRP) [10]. This hybrid approach has the advantage of significantly reducing the amount of control messages, since updated topology information must be maintained only among the nodes within each cluster. The main disadvantage is that a significant delay is introduced when a packet is destined to a node that belongs to a different cluster, since a new path should be searched in a reactive way.

Some assumptions about nodes' mobility in the network are at the basis of all these approaches, and these assumptions allow us to optimize the tradeoff between reducing the overhead, in terms of control messages exchanged in the network, and having a perfect knowledge of the network topology, which allows the design of optimal traffic routes, but at the cost of more frequent control messages. While it is possible to find an optimal tradeoff depending on the network mobility, it is challenging to deal with a network whose mobility characteristics are unknown, or can change over time. As an example, think of a crowd of people attending a sporting event.

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

The local topology among smartphones during the event is almost fixed, while it changes rapidly as soon as the event is finished and people begin moving out of the stadium.

In order to adapt to the changing mobility characteristics of such a network, we propose to physically divide the data plane (DP), including all the data traffic exchanged locally among the nodes, and the control plane (CP), comprehending all the control packets and the local routing decisions. This logic is borrowed from the software-defined network (SDN) paradigm [11], which is used in wired networks to provide centralized control over packet routing, even though the network topology does not include a central unit that is receiving and redirecting all the traffic.

In this paper, we investigate the main issues involved in realizing an SDN architecture for MANETs with real devices, and we propose a practical implementation of the SDN paradigm with devices running a Linux operating system (OS). In our devices, we have two wireless interfaces for the DP and the CP. For the CP, we build a star topology in which each node is communicating directly to a centralized unit (CU) that is making routing decisions in a centralized way, having full knowledge of the topology almost in real time. We assume that the cost per bit in this centralized network is high, e.g., in terms of energy or actual cost, so we limit the traffic in the CP to control packets. The DP is, instead, built on an ad hoc network, where each node can communicate only to nodes in close proximity. The cost per bit in this network is very low.

This framework has two main advantages. 1) The overhead introduced in the ad hoc network is minimal, since each node just needs to be aware of its neighbors and communicate this information to the CU via CP, and 2) the complexity at each node in the network is radically reduced, because the routing algorithm is run at the CU.

This work can be used as a starting point for the implementation of a wireless SDN testbed, providing a general SDN architecture and practical suggestions on how to implement such architecture while taking into account the limitations and constraints of commercial devices. The main contributions of this paper are summarized as follows.

- After describing the related works in Sec. II, in Sec. III we provide the logical details of each component of our SDN MANET architecture, explaining the reason for each design choice we made, based on the constraints imposed by the Linux operating system (OS) running in the devices.
- Our implementation of the SDN architecture is described in Sec. IV, with particular attention paid to the design constraints imposed by a MANET topology; the code used for our SDN MANET implementation is referenced in the paper, and the source code provided by us is available in [12] for the wireless networking community.
- In Sec. V, we highlight the advantages of our approach compared to a distributed state-of-the-art approach with OLSR; the results are obtained from a real network implementation with physical devices. Finally, Sec. VI concludes the paper.

II. Related Works

The SDN paradigm was originally designed for wired networks, where the topology is not changing and the channels are reliable. One of the first wireless SDN testbed was proposed in [13] using Wi-Fi routers and evaluating the performance in the occurrence of handovers between Wi-Fi and WiMAX.

Another practical implementation of the SDN paradigm in a wireless mesh network was implemented using NOX controllers [14]. In this paper, the CP and the DP use two separate service sets (SSs) with the same IEEE 802.11 interface. The OpenFlow control messages are exchanged with one SS identifier (SSID), while data packets are transmitted using a different SSID.

A hybrid approach is proposed in the OLSR-to-OpenFlow (O2O) architecture [15]. According to this approach, a node in the network is elected to be the controller. OLSR is used in the initial phase when all the topology and channel information is sent to the controller. The routing decisions for the whole network are then made at the controller with OpenFlow, and the routing rules are distributed to all the nodes in the network. If the topology changes, OLSR is used again to send the updated network information to the controller.

The main advantage of the O2O approach is the fact that it can implement a centralized routing algorithm, since all the network information is available at the controller node. The main drawback is that network updates, when there are topology changes, are extremely slow, because they require the collection of a lot of information from the network.

In our approach, by exploiting the two wireless interfaces, we maintain the advantages of O2O while providing a solution to the additional delay, since our CP is organized in a star (one-hop) topology.

III. Architecture of the SDN MANET

In this section, we present the designed architecture of our SDN MANET. In the DP, we just need to select a proper MAC layer that allows us to build an ad hoc network. For the CP, we should select 1) the SDN protocol, which allows the exchange of all the needed control packets; 2) the wireless switch, which allows each node to select the next hop in the multi-hop topology; and 3) the controller, which is located at the CU, receives all the topology information and runs the routing algorithm. In the following, we describe all these components.

A. MAC Protocol for DP

The MAC protocol for the DP should act in unlicensed bands without a centralized coordination[1]. A natural choice would be the carrier sense multiple access with collision avoidance (CSMA/CA) protocol, i.e., the standard IEEE 802.11 network. The problem is that the standard IEEE 802.11 is in infrastructure mode, thus requiring the existence of an AP that will receive and re-forward the packets, while it is not designed for a peer to peer (P2P) network, such as the one in our scenario. The IEEE 802.11s (Open Mesh Wi-Fi) can not be used either, since its MAC layer has a built-in switching

[1]The CU controls the routing of the packets, but it can not control in real time the medium access.

mechanism, that will not allow the SDN controller to have full control over the routes.

Instead, we have adopted the IEEE 802.11 P2P mode independent basic service set (IBSS) [16], which is the ad hoc mode standard of IEEE 802.11, giving a full control over the switching capabilities to the upper layers, as needed by our approach.

B. SDN Protocol

The SDN protocol should define a set of application program interfaces (APIs) that allow the exchange of control packets between the CU, which acts as a controller, and the network nodes. Among the several SDN protocols available in the literature [17]–[23], Open Flow protocol [24] is very popular, and it has been widely deployed particularly in data centers. There are several advantages of OpenFlow compared to other competing protocols.

First of all, it provides access to the parameters at the MAC layer, the network layer (NET), and the transport layer (TSP), as well as built-in methods to modify all of these parameters. This flexibility is particularly useful in order to optimize the network if a cross-layer controller is used.

Another advantage of OpenFlow is the presence of several open source software switch packages that support it, e.g., [25], [26].

C. Wireless Switch for Multi-hop

Our goal is to realize a multi-hop network in which we can control the routing with a centralized controller through the SDN architecture. In a Linux system, there are two main approaches to modify the routing paths in a network. The first approach (A1) is based on a direct modification of the routing table of each node. This requires a protocol that modifies the NET layer in the Linux kernel based on the information received from the SDN application running in the node. The second approach (A2) is based on the use of an SDN module (a wireless switch), which is the software component on the top of the Linux kernel that can put into action the routing decisions made at the SDN controller, i.e., it actually appends the new destination address to each packet.

We have chosen (A2) for its compatibility with the previous SDN framework, in order to provide a more general testbed for the networking community. Among the supported wireless switches, we select Open vSwitch (OVS) [25] and Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD) switch [26]. Both provide the needed functionalities in our scenario, but we implement OVS, because it has a simpler architecture, compared to CPqD, and it supports more versions of the Linux kernel, allowing further flexibility in the choice of node devices.

D. SDN Controller

The SDN controller is a software component installed in the CU, which is responsible for the management of the MANET, and in particular for the routing decisions. The SDN controller works as follows. First, it requests each OpenFlow switch (by means of OpenFlow rules) to send out link layer discovery packets (LLDPs) for neighbor discovery and to forward all the received LLDPs to the controller. In this way, the SDN controller has a full (and updated) knowledge of the network topology. Then, on the basis of the topology, the controller decides all the routes between each pair of nodes using a centralized topology-based routing algorithm.

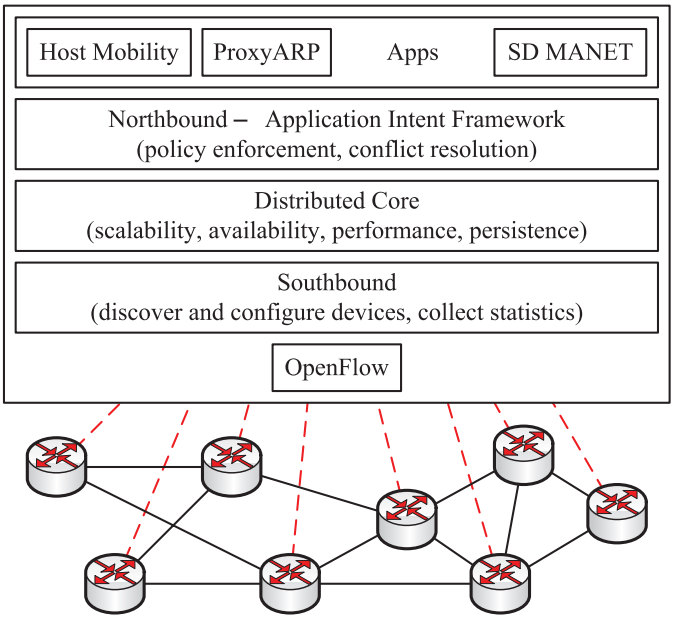

Fig. 1: The architecture of our ONOS controller.

There are several options available for SDN controller compatible with OpenFlow [14], [27]–[30], which differ based on the routing protocols implemented. Among the controllers supported by the developer community, Open Network Operating System (ONOS) [31] developed by ON.Lab was chosen because it provides a large number of resources that can contribute to this project.

In Fig. 1, we show the architecture of ONOS. The network information, in the form of link information, is collected by OpenFlow and communicated to the distributed core of ONOS through the southbound connections. The distributed core is responsible for translating this information into a format that can be processed by the routing algorithm. This new information is then sent up (through the northbound connections) to the application where the routing algorithm is running. There are several available examples of applications, including the Border Gateway Protocol (BGP) [32] and the Open Shortest Path First (OSPF) [33]. In general, any type of centralized routing algorithm can be implemented here.

IV. Implementation of a Multi-hop SDN MANET

In this section, we provide the implementation details to realize the SDN MANET, whose architecture is detailed in Sec. III. In particular, we overview the platform chosen to implement each SDN node, and provide the details of the modifications needed in the Linux kernel of each node. Then, we discuss the effects of the SDN framework on the data

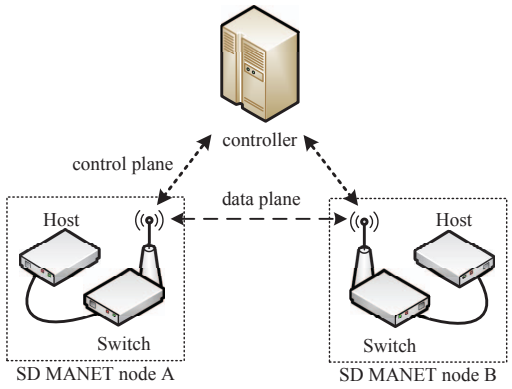

Fig. 2: Configuration of the SDN node in a MANET.

packets in the multi-hop SDN network, showing step by step all the modifications in the packet header.

A. Platform

The choice of a platform to run all the SDN code is limited to devices running the Linux OS, for compatibility with OVS. The choice of Linux OS is also motivated by the fact that any solution implemented in a Linux device can also be ported with minimal effort to other Linux-based systems, e.g., Android smartphones or tablets.

Among the available devices, we have selected the Raspberry Pi (RPi) Model B+. The reasons behind our choice are explained as follows. 1) In order to provide a useful tool for the networking community, we selected RPi because it has a large and active community of developers who will provide fast answers to any software questions. 2) Thanks to the active developers community, the most recent Linux kernel version is readily available for RPi. 3) The reduced cost of these devices is a great advantage for developers planning to build a large SDN testbed.

In Fig. 2, we show a scheme of the implementation of our SDN node. In the figure, we observe one SDN node, which is composed of two devices, a host and a switch, connected by an Ethernet cable[2]. The switch, in particular, is responsible for managing the two wireless interfaces towards the controller (as part of the CP) and towards the other SDN nodes in the network (as part of the DP, i.e., the MANET).

B. Linux Kernel

In order to implement the SDN architecture, we need to modify the Linux kernel of the switch device to allow the direct control from the CU of the node's routing. The main modifications are illustrated in Fig. 3.

In Fig. 3-(a) we show the logical data flow in the standard Linux kernel protocol stack from the physical (PHY) to the application (APP) layer, without any modification. First, we focus our attention on the data flow of the Ethernet frames, on the leftmost part of the figure. The Ethernet frames arrive to

[2]The idea of using two separate devices as a host and a switch makes the implementation and the debugging easier. In the future, we plan to implement both these logical parts into a single node.

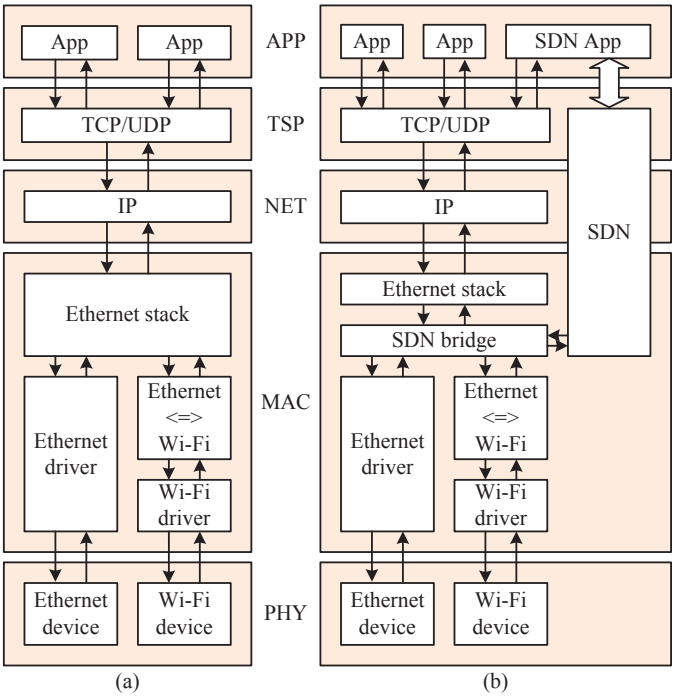

Fig. 3: Data flow in the Linux kernel network stack: (a) without the SDN modules; (b) after SDN modules are attached.

the device from copper cables (at the PHY layer) and are sent to the Ethernet driver (at the MAC layer), where their integrity is checked before removing the PHY header and sending them to the Ethernet stack. In the Ethernet stack, the MAC header is also removed before sending the packets up to the NET layer, where the routing is located.

In cases where a Wi-Fi packet is received from the wireless interface, the packet is first sent to the Wi-Fi driver where the PHY header is removed, then it is processed by a specific module that transforms the Wi-Fi packet into an Ethernet packet, which is subsequently sent to the Ethernet stack.

In Fig. 3-(b) we show the logical data flow including the new modules that are inserted in the Linux kernel to allow the SDN controller to modify the routing behavior at each node. In particular, the SDN bridge is a new module between the Ethernet/Wi-Fi drivers and the Ethernet stack, which can send incoming datagrams to the SDN block for further processing.

In the SDN block, the datagram can be modified according to the received rules from the SDN controller. In particular, in order to control the routing path, the rules imposed by the SDN controller involve the modification of the MAC header of the corresponding datagram. After these modifications, the datagram will be sent back to the SDN bridge, and forwarded to the Ethernet stack for the same processing as in the case of Fig. 3-(a). All the new SDN modules provided in this work can be downloaded in [12].

We highlight here the fact that the SDN rules are communicated to the SDN block by a SDN application (running in the APP layer). This application directly communicates with the controller in the CU via the CP using the same protocol stack, but a different wireless interface with respect to the DP packets.

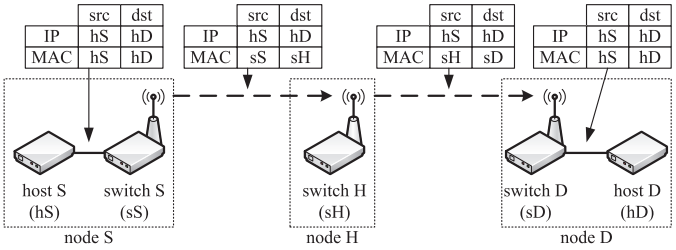

Fig. 4: Modifications in the MAC header for a packet generated at node S, relayed by node H, and destined for node D. Wired connections are denoted by solid lines, whereas wireless connections are represented with dashed lines.

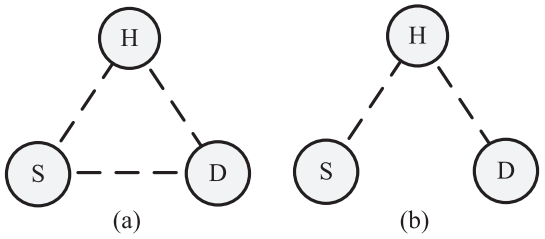

Fig. 5: Network topology for testing the throughput: (a) both a one-hop link and a two-hop link are available between S and D; (b) only the two-hop link is available.

C. SDN Multi-hop Network in Practice

In order to explain how each packet in a multi-hop network is interpreted and modified by the SDN framework, we provide a clarifying example, shown in Fig. 4, involving a two-hop transmission from a source node (S) towards a destination node (D). The packet is generated by S. It is first transmitted to a helper node (H), and then it is relayed by H to D. In this example, there is no direct connection between S and D. Furthermore, the address resolution protocol (ARP) information is provided directly by the ProxyARP application running on the SDN controller at the CU, which is in direct communication via CP with all the nodes in the network.

The packet is generated by host S (hS), which specifies in the IP header and in the MAC header the address of the destination, host D (hD). hS is not aware of any routing information needed to reach hD. It just transmits the packet through a wired channel to the switch of node S (sS), which is responsible for transmitting it wirelessly. sS is running the SDN module, shown in Fig. 3, which modifies the addresses of the source and the destination at the MAC layer, according to the instructions received from the SDN controller. Source and destination in the MAC header become sS and sH (the switch of node H), respectively. Then, the packet can be sent by sS, and it can be received by sH. Since the destination MAC address in the packet is sH, the packet is further processed by sH, and an acknowledgment (ACK) is sent back to sS.

We stress the fact that, for most of the Wi-Fi adaptors, a packet is discarded in the firmware if the MAC destination does not match, so the modification of the MAC address is a necessary step to deliver a packet in a multi-hop network.

A similar procedure is executed by sH, according to the SDN rules received from the SDN controller. Also in this case, the destination MAC address is modified, and the packet can be forwarded to the switch of the destination node (sD).

Finally, sD modifies both the source and the destination MAC addresses, to hS and hD, respectively, according to another rule from the SDN controller. This procedure, called MAC recovery, is necessary in order for hD to recognize that the packet was sent by hS, without the need of any further information from the SDN framework.

V. SDN TESTBED: PERFORMANCE COMPARISON

In this section, we showcase a practical implementation of our SDN framework with commercial devices, we set up the network and we send some data traffic with a multi-hop topology. In order to show the potential advantages of our framework, we run the same network with a distributed routing protocol, and we show the benefits of the SDN framework in the case of sudden topology changes.

A. Network Setup

The experimental scenario is composed of an SDN MANET with three SDN nodes, labeled S, H, and D, as in the previous section, deployed as shown in Fig. 5. Each of these nodes is composed of one RPi Model B+ and one Wi-Fi adapter (Ralink RT5370 USB), and the transmissions in the DP use the IEEE 802.11g ad-hoc mode (on channel 6). The three SDN nodes are equipped with OVS-2.4.0, and are connected to the CU running ONOS (the chosen OpenFlow controller) and our MANET application [12].

A second network, named OLSR MANET, is set up for comparison using the same three nodes (S, H, and D) in the same location and with the same topology. In this second network, indeed, the nodes are not equipped with our SDN framework, but they are running a distributed routing strategy, OLSR. In particular, *olsrd-0.8.8*, which is provided by the HSMM-Pi Project [34], is installed in the three nodes. The parameters of OLSR are set up as follows. The hello message interval, the hello validity time, and the topology control message intervals are set to 10, 1, and 0.5 seconds, respectively.

In both networks, the data traffic is generated at node S using the traffic generator *iPerf3* [35], which creates a random TCP flow towards the destination, node D. The length of the experiment is N seconds, and time is divided into intervals of 1 second. For each interval τ_n, with $n = 1, \ldots, N$, the end-to-end throughput is measured in bit-per-second (bps) as:

$$T(\tau_n) = \frac{\texttt{TCP RWND} \times 8}{\texttt{RTT}}, \quad (1)$$

where TCP RWND is the average receiving window size of TCP session during interval τ_i, and RTT is the average round-trip time, i.e., the elapsed time between the transmission of the first bit of a TCP segment sent and the receipt of the last bit of the corresponding TCP acknowledge.

In order to compare the behavior of the SDN MANET and the OLSR MANET in the case of a sudden topology change, we alternate the fully connected topology, shown in Fig. 5-(a), with the multi-hop topology, without a direct connection between S and D, shown in Fig. 5-(b).

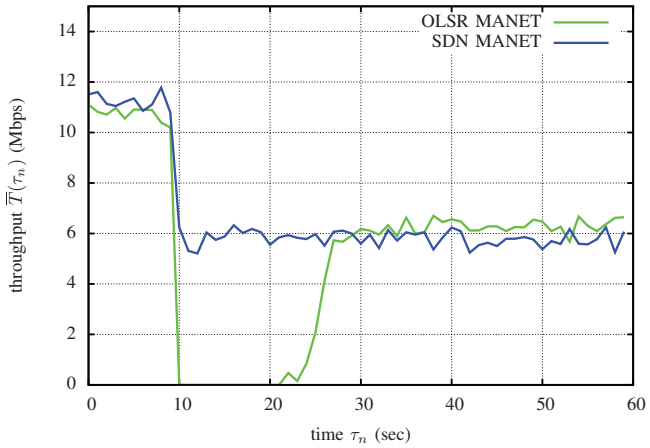

Fig. 6: Throughput as a function of time for SDN MANET and OLSR MANET for the "Link-down" experiment.

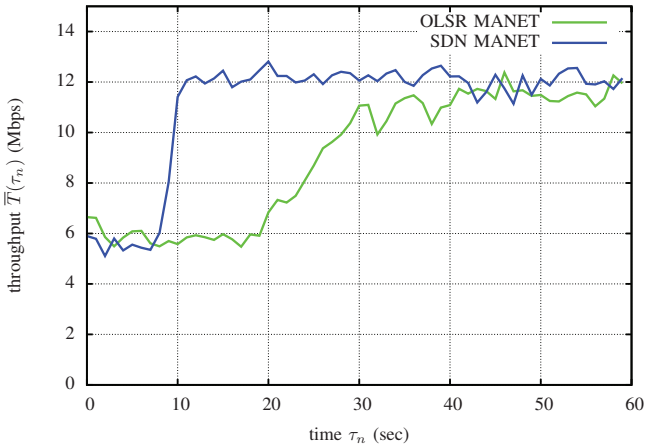

Fig. 7: Throughput as a function of time for SDN MANET and OLSR MANET for the "Link-up" experiment.

Since it is not easy to perfectly control a link failure by changing the location of the nodes, we emulate a link failure between S and D by designing a module at the MAC layer of the nodes that can reject all the packets coming from S (for node D), or from D (for node S). In this way, we can perfectly control in our experiment when the link between S and D fails, or when it is up again.

In the following, we describe three experiments that compare SDN MANET and OLSR MANET in the case of a change in the topology (due to the failure of an existing link, or to the restoration of a link previously non existing). Each experiment is repeated $M = 20$ times for both the SDN MANET and the OLSR MANET. The average throughput, shown in the results, is obtained as:

$$\overline{T}(\tau_n) = \frac{\sum_{m=1}^{M} T_m(\tau_n)}{M}\,, \tag{2}$$

where $T_m(\tau_n)$ is the throughput obtained during the time interval τ_n for the m^{th} experiment.

B. Link-down Experiment

In the first experiment in Fig. 6, we observe the effects of a link failure event, which changes the topology of our network. The experiment starts at $t = 0$ with the topology of Fig. 5-(a), where the three nodes (S, H and D) are fully connected. S starts sending TCP traffic towards D, and both the SDN MANET and the OLSR MANET choose the direct link between S and D.

Then, at time $t = 10$, the direct link between S and D fails, thus the topology becomes the one in Fig. 5-(b). The SDN controller is immediately notified about this event, and it promptly reacts, imposing a new SDN rule to nodes S and H. In this way, node S sends all the packets destined to D towards H, and H forwards these packets towards D. The throughput for SDN MANET is immediately restored to half the value of the initial throughput, since the new path from S to D now has two hops.

The OLSR MANET is able to identify the link failure and react to it by changing the path to D only at $t = 25$, with a delay of about 15 seconds, causing a significant throughput outage. This result is expected since OLSR has a fully distributed routing algorithm, which takes significant time to update. On the other hand, SDN MANET can exploit the CP, which allows the routing algorithm to be run in a centralized fashion at the CU, where all the information about link conditions are promptly collected.

C. Link-up Experiment

In the second experiment, in Fig. 7, we observe the averaged throughput experienced by SDN MANET and OLSR MANET when the initial topology is the one in Fig. 5-(b), i.e., a two-hop path between S and D. At $t = 10$ the direct link between S and D is also activated, as in Fig. 5-(a). We observe, as expected, that in the case of SDN MANET the network is able to react promptly to the change in topology, and the throughput almost doubles for $t > 10$. On the other hand, OLSR MANET has a delay of about 20 seconds before it is able to fully use the direct link, reaching the maximum throughput.

D. Fast-changing Topology Experiment

In the third experiment, in Fig. 8, we have a series of consecutive topology changes. At $t = 0$, the topology is the one in Fig. 5-(a) (direct link between S and D), then at time $t = 30$ the topology becomes the one in Fig. 5-(b) (two hops), then it switches again to Fig. 5-(a) at $t = 60$ and finally to Fig. 5-(b) at $t = 90$.

Also in this case, the experiment is repeated 20 times and the results are averaged over all the trials. For each topology change, we observe how the SDN MANET is able to react almost immediately to the topology change, while OLSR MANET reacts to the changes with a certain delay, causing a significant throughput loss, as expected.

VI. Conclusions

In this work, we presented a practical implementation of a SDN MANET that provides all the advantages of D2D data transmissions and, at the same time, has the flexibility of a

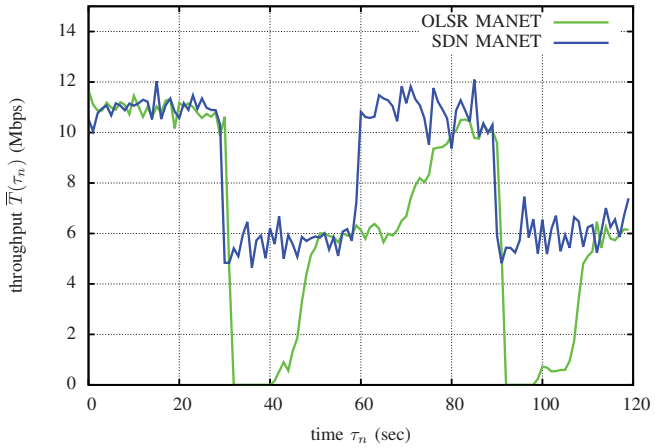

Fig. 8: Throughput as a function of time for SDN MANET and OLSR MANET for the "Fast-changing Topology" experiment.

centralized network management. We described the details of the SDN architecture, and we overviewed and referenced all the software components that we adopted. We contributed to this effort by providing new components, which are available to download in [12] for the wireless networking community.

In order to show the advantages of the SDN MANET, as well as the validity of all the software provided, we showed a comparison of our SDN MANET with an ad hoc network managed in a distributed way. We highlighted, with few simple examples, the significant advantages of our approach, in particular for a fast changing network topology.

In a future work, we plan to deal with a large scale SDN MANET, addressing the scalability issues that will likely rise up. In particular, we plan to collaborate with other interested national and international research institutions to build a large and distributed testbed, with which we can address several research problems, such as how to facilitate the coexistence of D2D communications in licensed bands [36].

ACKNOWLEDGMENTS

This work was partially supported by the Qualcomm Institute, University of California San Diego, under grant number 20134102.

REFERENCES

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2014-2019," May 2014.
[2] T. Aktas, G. Quer, T. Javidi, and R. Rao, "From Connected Vehicles to Mobile Relays: Enhanced Wireless Infrastructure for Smarter Cities," in *IEEE GLOBECOM*, 2016.
[3] I. Pappalardo, G. Quer, B. D. Rao and M. Zorzi, "Caching strategies in heterogeneous networks with d2d, small bs and macro bs communications," in *IEEE ICC*, May 2016.
[4] B. Kaufman and B. Aazhang, "Cellular networks with an overlaid device to device network," in *Asilomar Conference on Signals, Systems and Computers*, Oct. 2008, pp. 1537–1541.
[5] S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," RFC 2501, Jan. 1999.
[6] D. A. Rosen, F. J. Mammano, and R. Favout, "An electronic route-guidance system for highway vehicles," *IEEE Transactions on Vehicular Technology*, vol. 19, no. 1, pp. 143–152, Feb. 1970.
[7] M. Danieletto, G. Quer, R. R. Rao, and M. Zorzi, "CARMEN: a cognitive networking testbed on android OS devices," *IEEE Communications Magazine*, vol. 52, no. 9, pp. 98–107, Sept. 2014.
[8] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561, July 2003.
[9] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626, Oct. 2003.
[10] Z. J. Haas, "A new routing protocol for the reconfigurable wireless networks," in *Universal Personal Communications Record*, vol. 2, Oct. 1997, pp. 562–566 vol.2.
[11] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, July 2013.
[12] "SDN MANET application for ONOS." [Online]. Available: https://github.com/chinghanyu/onos-wfwd
[13] K.-K. Yap, R. Sherwood, M. Kobayashi, T.-Y. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar, "Blueprint for Introducing Innovation into Wireless Mobile Networks," in *ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, 2010, pp. 25–32.
[14] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an Operating System for Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, July 2008.
[15] A. Detti, C. Pisa, S. Salsano, and N. Blefari-Melazzi, "Wireless Mesh Software Defined Networks (wmSDN)," in *International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2013, pp. 89–95.
[16] T.-H. Lai and D. Zhou, "Efficient,and scalable IEEE 802.11 ad-hoc-mode timing synchronization function," in *Advanced Information Networking and Applications*, Mar. 2003, pp. 318–323.
[17] Cisco, "OpFlex: An Open Policy Protocol White Paper," Oct. 2014.
[18] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," RFC 7348, Aug. 2014.
[19] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 3920, Oct. 2004.
[20] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "The Locator/ID Separation Protocol (LISP)," RFC 6830, Jan. 2013.
[21] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification," RFC 5810, Mar. 2010.
[22] T. V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, "The SoftRouter Architecture," in *In ACM HOTNETS*, 2004.
[23] M. Caria, A. Jukan, and M. Hoffmann, "SDN Partitioning: A Centralized Control Plane for Distributed Routing Protocols," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 381–393, Sept. 2016.
[24] 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, Mar. 2008.
[25] "Open vSwitch." [Online]. Available: http://openvswitch.org/
[26] "CPqD - An OpenFlow 1.3 switch." [Online]. Available: https://github.com/CPqD/ofsoftswitch13
[27] "POX." [Online]. Available: http://www.noxrepo.org/pox/about-pox/
[28] "Beacon." [Online]. Available: https://openflow.stanford.edu/display/Beacon
[29] "Floodlight." [Online]. Available: http://floodlight.openflowhub.org/
[30] "Ryu." [Online]. Available: https://osrg.github.io/ryu/
[31] "Onos - open network operating system." [Online]. Available: http://onosproject.org/
[32] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006.
[33] J. Moy, "OSPF Version 2," STD 54, Apr. 1998.
[34] "Project HSMM-Pi." [Online]. Available: https://github.com/urlgrey/hsmm-pi
[35] "iPerf - The ultimate speed test tool for TCP, UDP and SCTP." [Online]. Available: https://iperf.fr/
[36] F. Librino and G. Quer, "D2D Communications in the Uplink: a Context-Aware Approach with Punishment," in *Proc. IEEE GLOBECOM*, Washington, DC, USA, Dec. 2016.