Design of Medium-Range Outdoor Wireless Mesh Network with Open-Flow Enabled Raspberry Pi

Soe Ye Htet[1], Khamxay Leevangtou[2], Phyo May Thet[3], Kiattikun Kawila[4], Chaodit Aswakul[5]
Wireless Network and Future Internet Research Unit,
[1,2,3,5]*Department of Electrical Engineering,*
[4]*Department of Computer Engineering,*
Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
[1]soe.y@student.chula.ac.th, [2]khamxay.l@student.chula.ac.th, [3]phyomaythet29@gmail.com,
[4]kiattikun.kaw@student.chula.ac.th, [5]chaodit.a@chula.ac.th

Abstract

The weakness of conventional wireless mesh network is its difficulty in management and configuration adjustment because of the complex structure of wireless mesh network. To cope with such difficulty, researchers have proposed to use the feature of software defined networking with inherently enhanced network programmability. However, most software defined wireless mesh network works have been investigated merely in indoor or emulated environments. This paper is concerned with the design and testbed preparation for the medium-range outdoor wireless mesh network based on OpenFlow-enabled Raspberry Pi. Based on the constructed testbed, preliminary investigation on achievable one/two-hop performance in terms of throughput as well as round-trip time is herein reported and a potential future application towards a road network traffic monitoring is highlighted.

Keywords: Outdoor Wireless Mesh Network, software defined networking, Raspberry Pi

1. Introduction

Wireless mesh networks can be constructed by connecting the network nodes in an ad-hoc mode with a wireless interface e.g. based on the IEEE 802.11 standard. The mesh network is scalable because it can be easily extended without the requirement of infrastructure-based access points. The network must be designed by considering a topology robustness with self-healing characteristics as enabled by a proper mesh network routing. For instance, a wide area outdoor mesh network with the 30 wireless nodes with 802.11 standard and OLSR routing protocol has been successfully constructed and investigated [5]. However, such traditional wireless mesh networks with a distributed routing protocol is generally difficult to manage. This is because of the complex structure of wireless mesh topology and hence a manual configuration often required upon significant network routing upgrades. Distributed routing also suffers from the lack of the network global view, and this could raise an issue on routing protocol efficiency.

To cope with such difficulties, researchers have proposed to use the feature of software defined networking (SDN) with inherently enhanced network programmability [6]-[9]. Here, the control plane and data plane are separated by using the standard OpenFlow [1] protocol. Advantageously, the centralized SDN controller receiving all necessary signaling via the control plane would have the global view of wireless mesh network status. An enhancement can thus be expected by SDN.

However, software defined wireless mesh networks (SDWMNs) as investigated in [6]-[9] merely focus on indoor or emulated environments. The objective of this paper is thus to design an outdoor SDWMN as well as to validate the design by an actual testbed implementation in a real target usage environment. The testbed preparation in this research is concerned with the medium-range outdoor wireless mesh network based on OpenFlow-enabled Raspberry Pi as a mesh node, thanks to Raspberry Pi 3's cost-effectiveness and obtainable computational power within a compact form factor. The design intention is for studying the achievable link and path throughputs upon the medium achievable range of wireless connectivity based on off-the-shelf wireless ad-hoc link antenna. Our design is aimed at a potential future application towards a road network traffic monitoring, where each Raspberry Pi serves both as a wireless signal relay as well as a sensor e.g. by attaching with a small camera to monitor road traffic conditions.

2. OpenFlow-Enabled Raspberry Pi-Based Design of Medium-Range Outdoor Wireless Defined Mesh Network

An example of future usage scenario of proposed outdoor SDWMN is depicted in Fig. 1. The topology consists of 2 gateways and 6 mesh nodes. Each gateway is operated as a server to receive sensor data from mesh nodes and to push the data potentially into a data cloud. Here, a node running the gateway functionality can also run the SDN controller. Intel®

NUC7i7BNH is used as a gateway and a Raspberry Pi 3 model B+ with Quad-Core CPU and 1-GByte RAM is used as a mesh node. Both the gateway and mesh nodes run the Ubuntu mate operating system. In each gateway and mesh node, a dual-band EDUP EP-AC1605 Wi-Fi USB adapter with two omnidirectional antennas is installed. The maximum allowed transmission power for ISM-band 2.4 GHz and 5.5 GHz are 0.1 W and 1 W in Thailand respectively. These transmission power should allow a per-hop distance between 100 meters and 300 meters, which is regarded here as the medium range in this design.

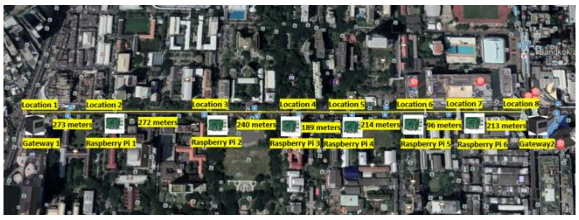


Fig. 1. Typical Installation Scenario of Outdoor SDWMN to Monitor Road Network Traffic (e.g. on PhayaThai Road, Bangkok).

The proposed SDWMN testbed has been designed with a typical usage deployment area as exemplified by the PhayaThai road segment in between Rama 1 road and Rama 4 road in Bangkok, for convenience of future installation and testing preparation. Here, there are five crossover bridges which are the suitable locations to place, wherever possible, the Raspberry Pi's in order to avoid the line-of-sight obstacles such as trucks or buses which can block the wireless signal. The average distance between adjacent crossover bridges is around 250 meters. An exception is on the distance between Raspberry Pi 3 and Raspberry Pi 5 over 400 meters which are not long enough to relay the signal from Raspberry Pi 3 to Raspberry Pi 5 directly. However, Raspberry Pi 4 can be placed on the side of the road in between Raspberry Pi 3 and Raspberry Pi 5.

In general, since a Raspberry Pi can be installed along either or both sides of the road (in addition to at the crossover bridge), a logical topology of the considered SDWMN scenario can be shown in Fig. 2. In this general topology of SDWMN, it is noticeable that each wireless link can suffer a difference level of reliability. For instance, the link in between the mesh nodes installed over the crossover bridge is expectedly more stable than the link in between the mesh nodes along the same side of the road. And the worst reliability can be expected from the particular wireless links that need to cross from one to the other side of the road regarding the mesh node installation locations. Over this general SDWMN case, a routing problem becomes non-trivial and hence the challenge in applying SDN especially in the real testbed environments.

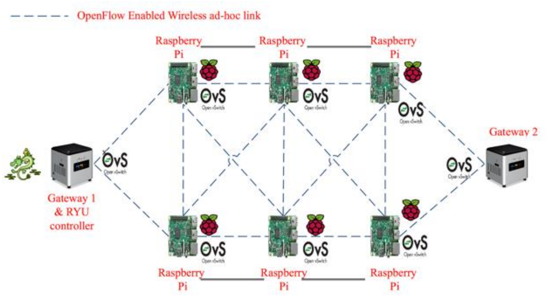


Fig. 2. Logical Topology of Outdoor SDWMN in Typical Road Network Traffic Monitoring over Bi-Directional Road Between Two Intersections.

Open Virtual Switch (OvS) [3] is installed in each mesh node and gateway to establish a connection between an SDN controller and the mesh nodes. OvS runs the standard OpenFlow protocol. A RYU controller [2] is chosen in this paper as the testbed's SDN controller. The RYU controller is Python-based and can support up to OpenFlow version 1.5 with usage convenience features e.g. GUI, OpenStack support, REST API. RYU controller has been suggested as a good choice for small businesses and research applications [10].

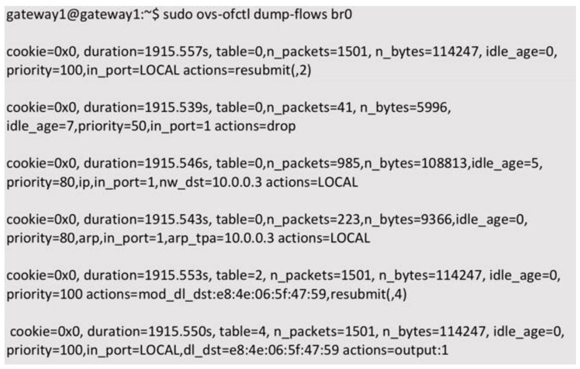
For the implementation of the control plane for SDWMN, there is a challenge in that most of the mesh nodes cannot reach the gateway directly in one wireless hop. This is unlike the wired SDN, whereby a direct communication channel is easily dedicated for the establishment of a control interface. For SDWMN, there are two possible options to implement the control plane i.e. with in-band control and out-of-band control.

Out-of-band control requires the separately dedicated control network and data network. Since the proposed outdoor SDWMN testbed is based on the IEEE 802.11 standard, at least two USB Wi-Fi adapters would be required in each of mesh nodes and gateways if the design is based on the out-of-band control approach. An extra hardware cost for control network is reducible in in-band SDN approach, whereby the control and data planes are implemented within the same physical interface at each node.

In this paper, therefore, the in-band SDN approach is used in order to reduce the hardware cost. In-band control plane is here established by installing our properly defined forwarding rules to the OvS of each mesh node to relay address resolution protocol (ARP) packets and transmission control protocol (TCP) packets between the mesh node and the SDN controller. These forwarding rules have been pre-installed at the mesh node as a script that will be automatically executed every time that the mesh node is restarted.

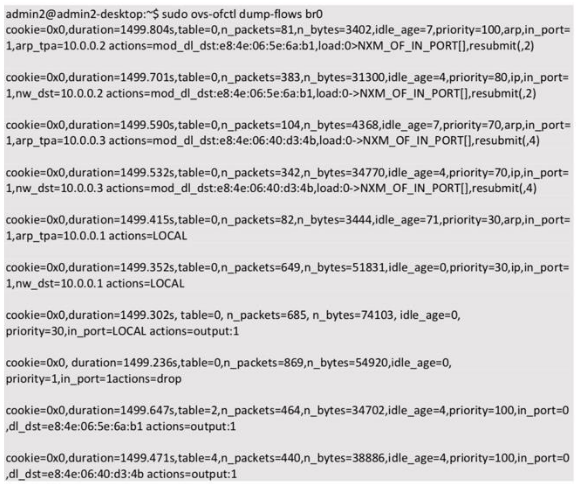
3. Preliminary Testbed Testing

From Fig. 1, as a preliminary testing, a gateway has been installed at location 2, and two Raspberry Pi's (Raspberry Pi 1 and Raspberry Pi 2) have been installed at locations 3 and 4. Fig. 3 shows the OvS forwarding rules which enable as the in-band network in this preliminary testing. Fig. 3 (a) is the forwarding rules for the gateway, Fig. 3 (b) and (c) are the forwarding rules for mesh nodes (Raspberry Pi 1 and Raspberry Pi 2). Among these figures, the forwarding rules in Fig. 3 (b) define the forwarding rules to enable relay function of wireless relay node between Gateway and Raspberry Pi 2. Here, the IP addresses of gateway, Raspberry Pi 1 and 2 are 10.0.0.3, 10.0.0.1 and 10.0.0.2, respectively. Additionally, the MAC addresses of gateway, Raspberry Pi 1 and 2 are e8:4e:06:40:d3:4b, e8:4e:06:5f:47:59 and e8:4e:06:5e:6a:b1, respectively.



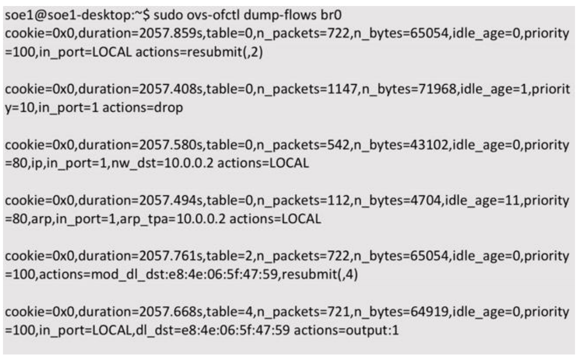
```
gateway1@gateway1:~$ sudo ovs-ofctl dump-flows br0

cookie=0x0, duration=1915.557s, table=0,n_packets=1501, n_bytes=114247, idle_age=0,
priority=100,in_port=LOCAL actions=resubmit(,2)

cookie=0x0, duration=1915.539s, table=0,n_packets=41, n_bytes=5996,
idle_age=7,priority=50,in_port=1 actions=drop

cookie=0x0, duration=1915.546s, table=0,n_packets=985,n_bytes=108813,idle_age=5,
priority=80,ip,in_port=1,nw_dst=10.0.0.3 actions=LOCAL

cookie=0x0, duration=1915.543s, table=0,n_packets=223,n_bytes=9366,idle_age=0,
priority=80,arp,in_port=1,arp_tpa=10.0.0.3 actions=LOCAL

cookie=0x0, duration=1915.553s, table=2, n_packets=1501, n_bytes=114247, idle_age=0,
priority=100 actions=mod_dl_dst:e8:4e:06:5f:47:59,resubmit(,4)

 cookie=0x0, duration=1915.550s, table=4, n_packets=1501, n_bytes=114247, idle_age=0,
priority=100,in_port=LOCAL,dl_dst=e8:4e:06:5f:47:59 actions=output:1
```

(a)



```
admin2@admin2-desktop:~$ sudo ovs-ofctl dump-flows br0
cookie=0x0,duration=1499.804s,table=0,n_packets=81,n_bytes=3402,idle_age=7,priority=100,arp,in_port=
1,arp_tpa=10.0.0.2 actions=mod_dl_dst:e8:4e:06:5e:6a:b1,load:0>NXM_OF_IN_PORT[],resubmit(,2)

cookie=0x0,duration=1499.701s,table=0,n_packets=383,n_bytes=31300,idle_age=4,priority=80,ip,in_port=
1,nw_dst=10.0.0.2 actions=mod_dl_dst:e8:4e:06:5e:6a:b1,load:0->NXM_OF_IN_PORT[],resubmit(,2)

cookie=0x0,duration=1499.590s,table=0,n_packets=104,n_bytes=4368,idle_age=7,priority=70,arp,in_port=
1,arp_tpa=10.0.0.3 actions=mod_dl_dst:e8:4e:06:40:d3:4b,load:0->NXM_OF_IN_PORT[],resubmit(,4)

cookie=0x0,duration=1499.532s,table=0,n_packets=342,n_bytes=34770,idle_age=4,priority=70,ip,in_port=
1,nw_dst=10.0.0.3 actions=mod_dl_dst:e8:4e:06:40:d3:4b,load:0->NXM_OF_IN_PORT[],resubmit(,4)

cookie=0x0,duration=1499.415s,table=0,n_packets=82,n_bytes=3444,idle_age=71,priority=30,arp,in_port=
1,arp_tpa=10.0.0.1 actions=LOCAL

cookie=0x0,duration=1499.352s,table=0,n_packets=649,n_bytes=51831,idle_age=0,priority=30,ip,in_port=
1,nw_dst=10.0.0.1 actions=LOCAL

cookie=0x0,duration=1499.302s, table=0, n_packets=685, n_bytes=74103, idle_age=0,
priority=30,in_port=LOCAL actions=output:1

cookie=0x0, duration=1499.236s,table=0,n_packets=869,n_bytes=54920,idle_age=0,
priority=1,in_port=1actions=drop

cookie=0x0,duration=1499.647s,table=2,n_packets=464,n_bytes=34702,idle_age=4,priority=100,in_port=0
,dl_dst=e8:4e:06:5e:6a:b1 actions=output:1

cookie=0x0,duration=1499.471s,table=4,n_packets=440,n_bytes=38886,idle_age=4,priority=100,in_port=0
,dl_dst=e8:4e:06:40:d3:4b actions=output:1
```

(b)



```
soe1@soe1-desktop:~$ sudo ovs-ofctl dump-flows br0
cookie=0x0,duration=2057.859s,table=0,n_packets=722,n_bytes=65054,idle_age=0,priority
=100,in_port=LOCAL actions=resubmit(,2)

cookie=0x0,duration=2057.408s,table=0,n_packets=1147,n_bytes=71968,idle_age=1,priorit
y=10,in_port=1 actions=drop

cookie=0x0,duration=2057.580s,table=0,n_packets=542,n_bytes=43102,idle_age=0,priority
=80,ip,in_port=1,nw_dst=10.0.0.2 actions=LOCAL

cookie=0x0,duration=2057.494s,table=0,n_packets=112,n_bytes=4704,idle_age=11,priority
=80,arp,in_port=1,arp_tpa=10.0.0.2 actions=LOCAL

cookie=0x0,duration=2057.761s,table=2,n_packets=722,n_bytes=65054,idle_age=0,priority
=100,actions=mod_dl_dst:e8:4e:06:5f:47:59,resubmit(,4)

cookie=0x0,duration=2057.668s,table=4,n_packets=721,n_bytes=64919,idle_age=0,priority
=100,in_port=LOCAL,dl_dst=e8:4e:06:5f:47:59 actions=output:1
```

(c)

Fig. 3. OvS Forwarding Rules of In-band at (a) Gateway (b) Raspberry Pi 1 (Wireless Relay Node) (c) Raspberry Pi 2.

Since EDUP EP-AC1605 is a dual-band antenna, 2.4 GHz band or 5.5 GHz band can be selected. The chosen testbed environment is located in an urban area which is not interference-free, with higher interference level in 2.4 GHz than in 5.5 GHz bands potentially because of nearby Bluetooth devices inside cars. Therefore, the 5.5 GHz band is expectedly more stable than the 2.4 GHz band and used in our test. Figs. 4 and 5 show the equipment of gateway and mesh node.

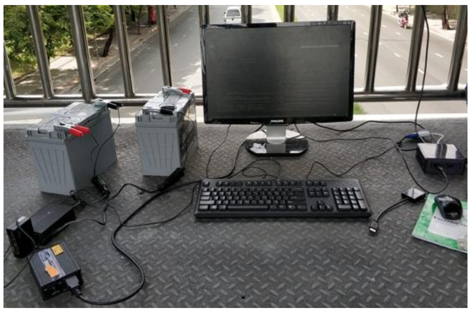


Fig. 4. Gateway Equipment.

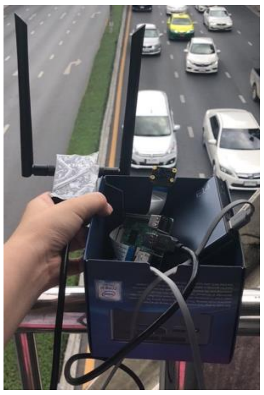


Fig. 5. Mesh Node Equipment.

Iperf software has been used to measure the TCP throughput of the wireless route with 1 and 2 hops. Table 1 reports the results of TCP iperf from 3 runs, each with the measurement period of 100 seconds. In order to measure the average round-trip time (RTT), a ping program is used. Here, 200 ICMP packets have been generated for each run and the average RTT value from each run is shown in Table 2.

Tables I and II confirm that the OvS forwarding rules of in-band control can provide effectively the necessary control plane for the implementation of data packet forwarding. In addition, based on the expected target usage scenario of a road network traffic monitoring, the reported round-trip time results confirm that the current network settings can be applied beneficially in the future large-scale road traffic monitoring system. In practice, even for an automatic control of traffic signal light based on wireless sensor network, a latency as high as a second should be acceptable. Future investigations are, however, needed to thoroughly confirm the usability of this SDWMN for that practical application.

Table. I. Average TCP Throughput.

	Average throughput for 1 hop (Mbit/sec)	Average throughput for 2 hops (Mbit/sec)
Test 1	10.9	7.1
Test 2	9.1	2.8
Test 3	10.7	2
Total average throughput (Mbit/sec)	10.2	3.9

Table. II. Average Round-Trip Time.

	Average RTT for 1 hop (ms)	Average RTT for 2 hops (ms)
Test 1	13.3	45.8
Test 2	9.0	46.8
Test 3	7.0	38.1
Total average RTT (ms)	9.8	43.6

4. Conclusion

In this paper, we have designed and implemented the preliminary testbed for measuring the network throughput and round-trip time performance of an outdoor medium-range SDWMN testbed. In-band control approach has been implemented successfully to save the extra hardware cost. The network covers only up to 2 hops simplified routing with 1 gateway.

Our ongoing work is to increase the number of mesh nodes in between the two gateways, where OpenFlow-based routing adaptation needs be verified. To enable the final SDWMN testbed robustness, a 24-hour test span is in plan, whereby the effect of other environmental conditions (e.g. ambient temperature, time of day) can be further studied.

References

[1] 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, pp. 69-74, 2008.

[2] RYU SDN Framework [Online] – Available from: https://osrg.github.io/ryu-book/en/Ryubook.pdf [April 20, 2018].

[3] Open vSwitch [Online] – Available from: http://openvswitch.org/ [April 10, 2018].

[4] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum and L. Viennot, “Optimized link state routing protocol for ad hoc networks,” IEEE International Multi Topic Conference, (IEEE INMIC 2001), pp. 62-68, 2001.

[5] D. Wu, D. Gupta and P. Mohapatra, "QuRiNet: A wide-area wireless mesh testbed for research and experimental evaluations," 2nd International Conference on COMmunication Systems and NETworks (COMSNETS 2010), pp. 1221-1237, 2010.

[6] A. Detti, C. Pisa, S. Salsano and N. Blefari-Melazzi, "Wireless mesh software defined networks (wmSDN)," IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 89-95, 2013.

[7] F. Yang, V. Gondi, J. O. Hallstrom, K.-C. Wang and G. Eidson, "OpenFlow-based load balancing for wireless mesh infrastructure," IEEE 11th Consumer Communications and Networking Conference (CCNC), pp. 444-449, 2014.

[8] W. J. Lee, J. W. Shin, H. Y. Lee and M. Y. Chung, "Testbed implementation for routing WLAN traffic in software defined wireless mesh network," 8th International Conference on Ubiquitous and Future Networks(ICUFN), pp. 1052-1055, 2016.

[9] P. Patil, A. Hakiri, Y. Barve and A. Gokhale, "Enabling software-defined networking for wireless mesh networks in smart environments," IEEE 15th International Symposium on Network Computing and Applications (NCA), pp. 153-157, 2016.

[10] O. Salman, I. H. Elhajj, A. Kayssi and A. Chehab, "SDN controllers: a comparative study," 18th Mediterranean Electrotechnical Conference (MELECON), pp. 1-6, 2010.