Design and Preliminary Functionality Test of Road Network Traffic Monitoring System Based on Indoor SDWMN In-Band Architecture

Phoo Phoo Thet Lyar Tun and Chaodit Aswakul

Chulalongkorn University, Thailand
phoophootltun@gmail.com, chaodit.a@chula.ac.th
Department of Electrical Engineering, Faculty of Engineering
Wireless Network and Future Internet Research Unit

Abstract. Software-defined wireless mesh network (SDWMN) combines the functionalities of wireless mesh network (WMN) and software defined networking (SDN) to achieve the goal of being effectively manageable of WMN. In this paper, the indoor SDWMN in-band testbed is proposed and implemented. The design and implementation of indoor SDWMN in-band testbed proposed in this paper is the preliminary testbed for the future real outdoor SDWMN in-band testbed for road traffic monitoring system. The testing results of indoor SDWMN in-band discussed in this paper shows that the system can function properly and becomes ready for future testing in the real outdoor environment.

Keywords: Software defined wireless mesh network, road traffic monitoring application,in-band

1 Introduction

Bangkok, the capital city of Thailand, the actual location of future testbed area, is ranked as the most congested city in Asia and the second most congested city globally [1]. To solve this problem of road traffic congestion, it is necessary to set up road traffic monitoring systems based on the various types of network infrastructures. Setting up the underlying network infrastructure requires two basic kinds of connectivities, namely wired connectivity and wireless connectivity. Wired connectivity to set up the network infrastructure costs a lot as the network becomes large. Therefore, the consideration of wireless connectivity is the best choice for this paper for cost-effectiveness. Among the wireless connectivities, in this paper, IEEE 802.11 standard Wi-Fi [2] connection is mainly used together with other wide coverage technologies e.g. 3G,4G connections, LoRaWAN [3], Zigbee [4] and WiMAX [5]. For the IEEE 802.11 wireless network setting, there are two modes of setting up the network, namely the infrastructure mode and the ad-hoc mode. In this work, the ad-hoc mode is chosen for the consideration of being easily extendable of the network.

The multi-hop ad-hoc network, the wireless mesh network (WMN) [6] is used for the cost-effectiveness of this paper. There are three traditional distributed routing protocols used in WMN, namely, proactive protocol, reactive protocol, and hybrid protocol [7]. All these traditional wireless ad-hoc routings are by-design distributed. Therefore, there is no central point to be controlled and managed. However, such distributed routing behavior is difficult to be modified.

Software Defined Netwoking (SDN) [8] is composed of the three planes, which are (i) data plane (ii) control plane and (iii) application plane. The forwarding elements are located on the forwarding plane. In SDN, the forwarding elements need to follow the instruction from the SDN controller for the behavior of packet forwarding, and the forwarding elements cannot decide for itself how to forward the packets. In traditional networking, the routing table inside the forwarding elements lets them decide how to forward the packets. In SDN, the SDN controller, which is the brain of the whole network, is located on the control plane. SDN controller uses OpenFlow protocol at the southbound interface as the communication protocol between the control and data planes. Thus, SDN with the help of OpenFlow protocol [9] has in the past been proposed to eliminate the limitation of traditional WMN, and the feature of network programmability provided by SDN lets the network administrator modify the routing behavior easily by using high-level programming languages. SDN based networks are easy to be controlled, altered, and managed because of the powerful features of separating the control plane and data plane of the network.

There are two ways to set up the control plane for the SDN, the out-of-band architecture uses the dedicated control plane and the in-band architecture uses the shared control plane [10]. Software-defined wireless mesh network (SDWMN) [11] combines the functionalities of WMN and SDN to achieve the goal of being effectively manageable of WMN and the cost-effective WMN has been implemented as a real outdoor testbed for the near real-time road traffic monitoring application using Apache-Kafka [12]. In the SDWMN in-band testbed proposed in [11], some mesh nodes has four hops far from the SDN controller. So, the control plane is unstable for this work. The design proposed in this paper concerns about the reliability of the control plane.

In the literature of wireless SDN [13–22], various approaches have been proposed and tested in the simulated, emulated and indoor environments. The researchers of [11] and [12] have tested in the outdoor environment for their road traffic monitoring application. The work of [11] and [12] uses the SDWMN in-band architecture. The controller placement of the work [11] is at the gateway of the network. In this paper, the indoor SDWMN in-band testbed is proposed and implemented. The design and implementation of indoor SDWMN in-band testbed proposed in this paper is the preliminary testbed for the future real outdoor SDWMN in-band testbed for road traffic monitoring system. The results of indoor SDWMN in-band discussed in this paper shows that the system is ready for testing real outdoor SDWMN in-band testbed.

The rest of the paper is summarized as follows. Section 2 presents the design and implementation of indoor SDWMN in-band Architecture. Section 3 gives the discussion based on the results received from the indoor SDWMN in-band testbed. Finally, the paper is concluded in Section 4.

2 Design of Indoor SDWMN In-Band Architecture

The design of SDWMN in-band testbed is based on the idea and work of [11]. However, the design proposed in this paper has the modifications on [11] such as the routing patterns and rerouting patterns of the control plane and data plane and controller placement so that the network can achieve an efficient control plane performance by reducing the hops between the controller and nodes. The road traffic monitoring application applied in the proposed testbed is based on the idea and work of [12].

2.1 Design of Indoor SDWMN In-Band Testbed

The indoor SDWMN in-band testbed is depicted in Figure 1. The topology consists of six wireless mesh nodes pi1, pi2, pi3, pi4, pi5 and pi6, two gateways gw1 and gw2, one Ryu controller, one Kafka external broker and one switch. This switch is used to connect between the two gateways, the Ryu controller and the Kafka external broker. Six raspberry pi's 3 model B [23] with Quad-Core CPU and 1-GByte RAM are used as the testbed's wireless mesh nodes. Intel R NUC7i7BNH [24] is used as the two gateways. Dual-band EDUP EP-AC1605 Wi-Fi USB adapter [25] with two omnidirectional antennas are attached to the mesh nodes and the gateways. Since the road traffic monitoring is applied to the indoor SDWMN-indoor testbed, it can be said that there are two layers of abstraction. First is the network layer that is SDWMN in-band layer, and the second is the application layer that the road traffic monitoring application is running on. Therefore, for example, the raspberry pi 3 model B is used as the wireless mesh node in the network layer and is used as the application layer's Kafka producer. Intel R NUC7i7BNH is used as the wireless gateways in the network layer, and it is used as the Kafka broker, Kafka consumer and Kafka producer in the application layer.All the software and hardware selection, the frequency band selection for the network, the location selected for the final outdoor testbed, the reason to choose whether in-band or out-of-band, and the road traffic monitoring application appliance in the testbed are based on the work of [11] and [12].

In [11], gateway gw2 has to pass through 4 hops to reach the Ryu controller, which is in gw1. By considering the control plane reliability improvement, in this design, the per-hop distance between the nodes and the Ryu controller is reduced. As shown in Figure 1, pi1, pi2 and pi4 are going to gw1, gw1, in turn, goes to the Ryu controller and pi3, pi5 and pi6 are going to gw2, gw2, in turn, goes to Ryu controller to establish the control plane. It means, in this design, we consider mainly for the controller placement and control plane reliability improvement. The data plane traffics are the images captured by the pi cameras. The data plane traffics use the same routing patterns as in [11] to communicate to the Kafka external broker. Therefore, in this design, the SDWMN in-band network is used to run the the road traffic monitoring application. The indoor SDWMN in-band testbed has been carried out in the author's room, which is located in the Ban Ratchathewi Apartment, Bangkok. Figure 2 and Figure 3 show the indoor equipment settings.

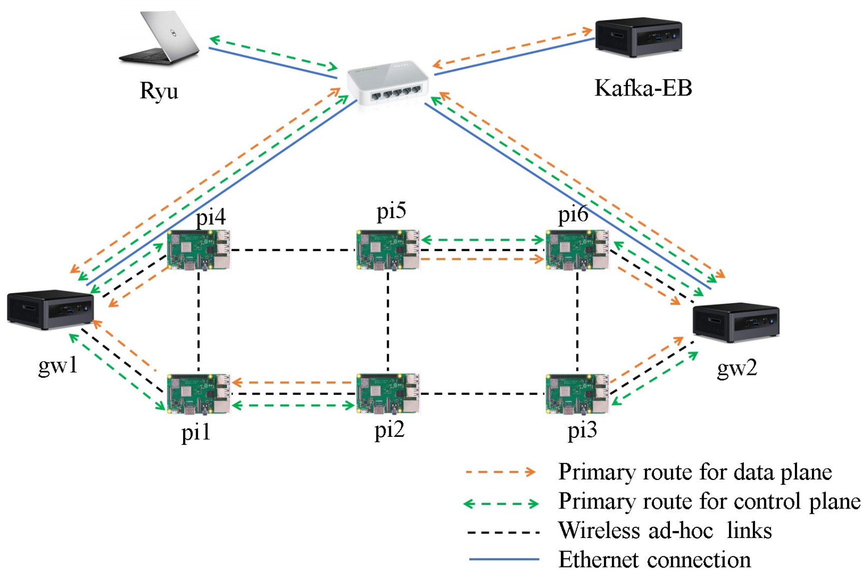


Fig. 1: Design of preliminary indoor SDWMN in-band testbed.

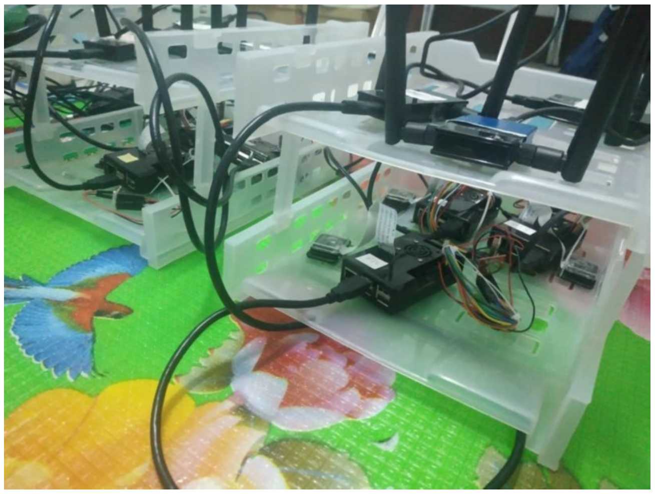


Fig. 2: Equipment setting of raspberry pi.

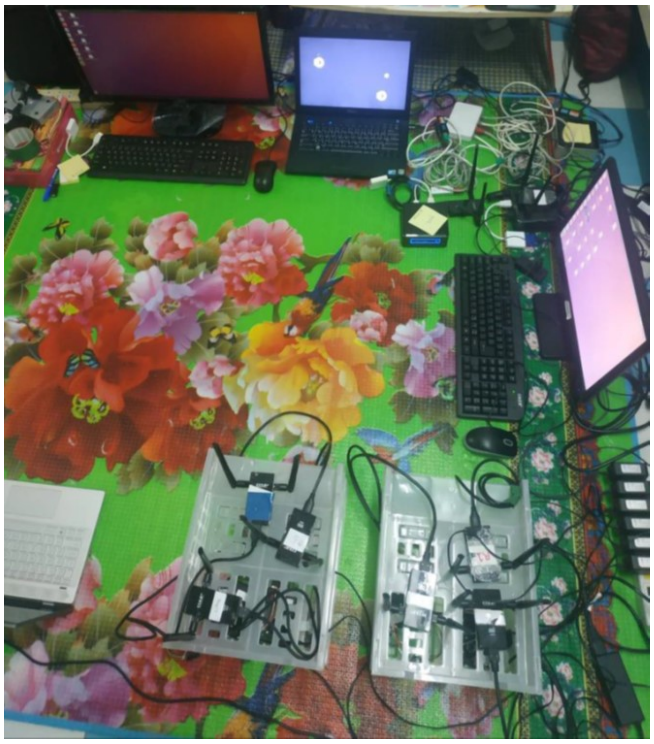


Fig. 3: Equipment setting of indoor SDWMN in-band testbed.

2.2 Implementation of Indoor SDWMN In-Band Testbed

For the physical implementation of indoor SDWMN in-band testbed, the devices listed in Table 1 are used. The software needed for installation are listed in Table 2. The primary route patterns and the rerouting patterns for the data plane and control plane of the SDWMN in-band tested are shown in Figure 4.

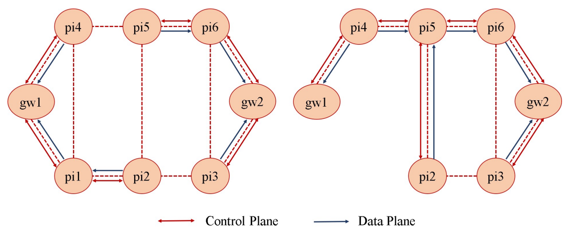
According to the design of the SDWMN in-band testbed presented in the previous section, the Ryu controller runs in the different devices. In contrast, in [11], the Ryu controller runs in the gateway. In this SDWMN in-band testbed, the two gateways, gw1 and gw2 in 1, serve as the relay functions for the control plane traffics and data plane traffics so that the mesh nodes can communicate to the Ryu controller and the Kafka external broker. Therefore, the gateway has two interfaces. The first one is the wireless ad-hoc interface to communicate with the wireless mesh nodes. The second one is the LAN interface to establish the LAN connection to the Ryu controller for the control plane traffic, and to the Kafka, external broker for the data plane traffics. All the necessary ideas, concepts, installation, and the implementation for the road traffic monitoring running on top of the SDWMN in-band testbed are the same as [12].

Table 1: Hardware list of indoor SDWMN in-band testbed.

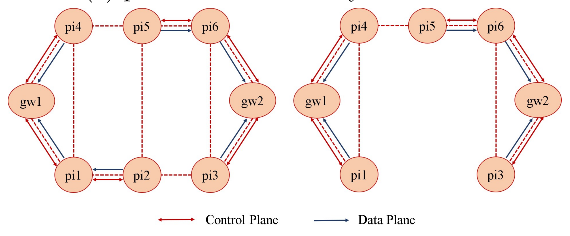
Device	Functionality	Quantity
Intel ® NUC7i7BNH	As wireless gateway (Kafka producer/broker/consumer)	3
Raspberry pi 3 model B	As wireless mesh node (Kafka producer)	6
EDUP EP-AC1605 dual-band antenna	For the network reachability	8
Raspberry pi camera module	To capture images	6
Switch	To set up LAN connection between gateways, Ryu controller and Kafka external broker	1
Laptop	To run Ryu controller	1

Table 2: Software list of SDWMN in-band testbed.

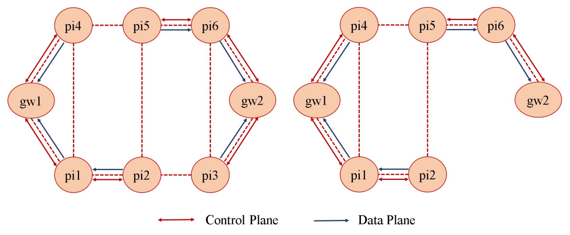
Software	Function	Installed node
Open Vswitch	Virtual OpenFlow switch	Gateways and wireless mesh nodes
RYU [27]	SDN controller	Dedicated laptop
Ubuntu Mate 16.04 (32 bit) [28]	Linux operating system	Wireless mesh nodes (Kafka producers)
Ubuntu (32 bit) [29]	Linux operating system	Gateways, Laptop for Ryu controller, Kafka external broker



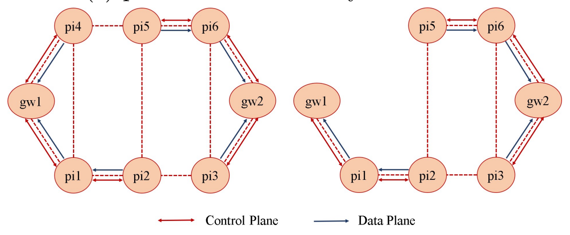
(a) pi1 unreachable to Ryu controller



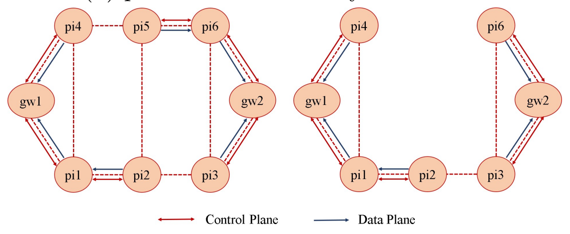
(b) pi2 unreachable to Ryu controller



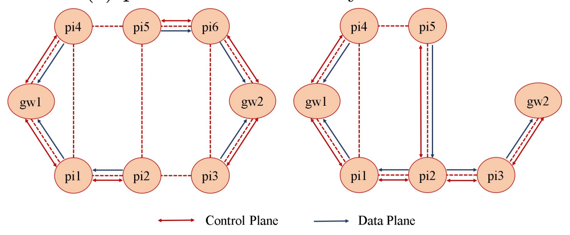
(c) pi3 unreachable to Ryu controller



(d) pi4 unreachable to Ryu controller



(e) pi5 unreachable to Ryu controller



(f) pi6 unreachable to Ryu controller

Fig. 4: Routing patterns and rerouting patterns of indoor SDWMN in-band testbed.

3 Functionality Testing of Indoor SDWMN In-Band Testbed

The indoor SDWMN in-band testbed applying road traffic monitoring application is tested to test the routing patterns and rerouting patterns of the data plane and control plane, there are a total of 7 scenarios. The testing scenarios range from when all pi's are reachable to Ryu controller, when pi1 is unreachable to Ryu controller, when pi2 is unreachable to Ryu controller, when pi3 is unreachable to Ryu controller, when pi4 is unreachable to Ryu controller, when pi5 is unreachable to Ryu controller and when pi6 is unreachable to Ryu controller. For each testing scenario, there are three sub-testings. These three sub-testings are aimed at testing for the data plane traffics, which are the images captured by the pi's cameras.in the preliminary indoor testing, the three sub-testings are when the pi's camera is covered by black paper, when the pi's camera is covered by white paper and when the pi's camera is uncovered.

The reason why the sub-testings are carried out is to test for the differences of the image size. Image size will be different according to the daytime, nighttime and traffic intensity when we test in outdoor testbed due to the compression used in [12]. The total size of the images produced from each pi can be captured at the gateways. The total size of the control plane traffics communicating between the pi's and the Ryu controller can be captured at the Ryu controller.in the preliminary indoor testing, the seven testing scenarios are tested separately. For each sub-testing of each testing scenario, it takes 3 minutes. The total control plane traffic testing also takes 3 minutes for each scenario. There are also seven testing scenarios for the control plane traffic testing.

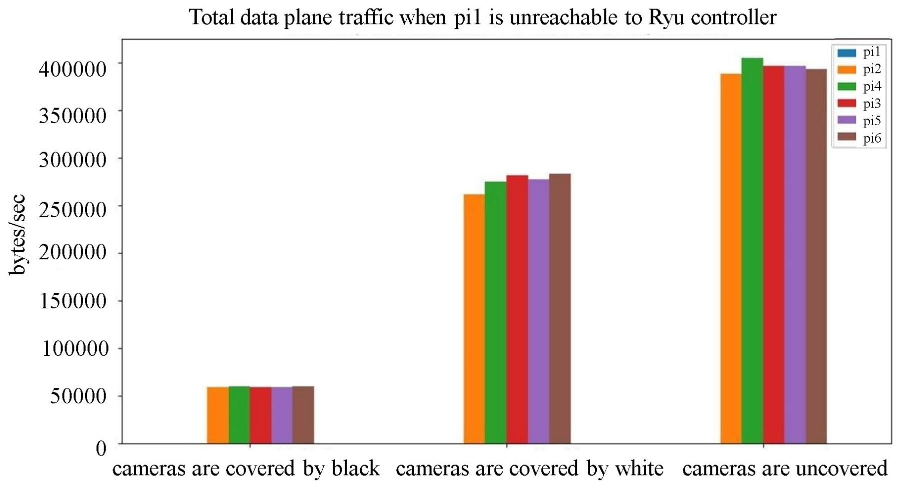


Fig. 5: Total data plane traffics from each pi when pi1 is unreachable to Ryu controller.

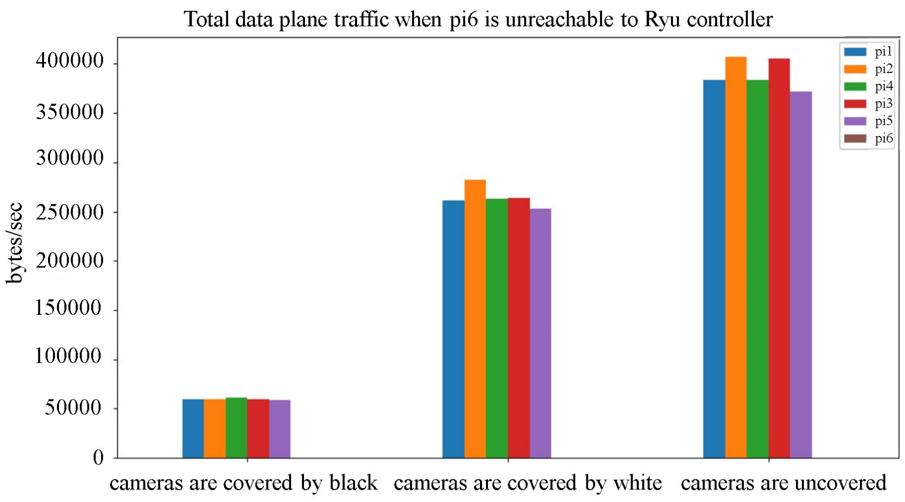


Fig. 6: Total data plane traffics from each pi when pi6 is unreachable to Ryu controller.

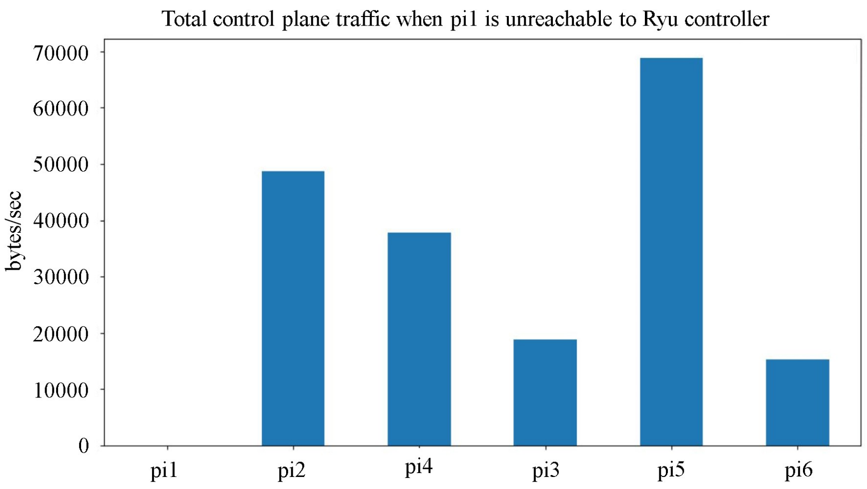


Fig. 7: Total control plane traffics from each pi when pi1 is unreachable to Ryu controller.

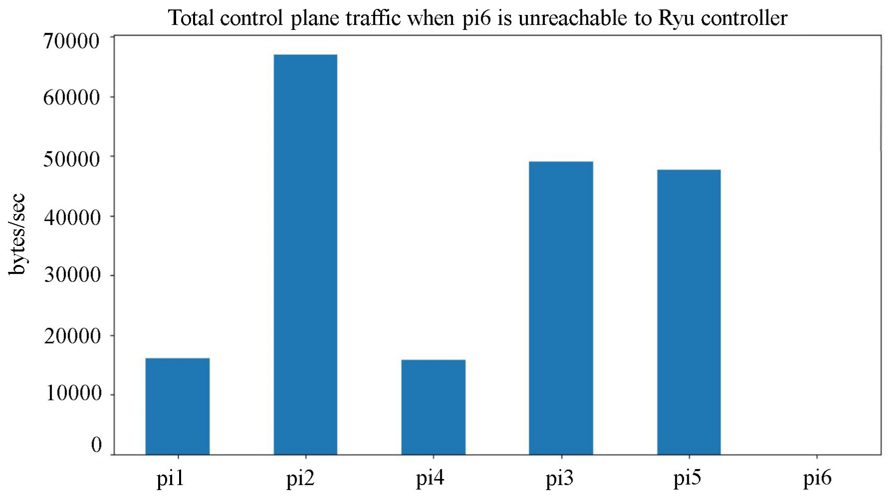


Fig. 8: Total control plane traffics from each pi when pi6 is unreachable to Ryu controller.

Even though pi1and pi6 are unreachable to Ryu controller, both the control plane traffic and data plane traffic of pi2 and pi5 are still reachable to the gateways as shown in Figures 5, 6, 7 and 8. Since the rerouting takes a few seconds with the same setting as in [11], the total data plane traffic from pi2 and pi5 is lower than in the case of pi1 and pi6 unreachable cases as presented in Figures 5 and 6. In the scenario of pi1 and pi6 unreachability, the Ryu controller has to assign the rerouting flow rules to pi2 and pi5 so that they can still establish the control plane as described in Figures 4a and 4f. Therefore, in Figures 7 and 8, the total control plane traffic from pi nodes are different compared to the other scenarios in which the total control plane traffic from pi nodes are almost the same.

4 Conclusion

In this paper, the design and preliminary functionality testing of SDWMN in-band testbed applying road traffic monitoring application is proposed. The proposed design in this paper aims to achieve the better control plane reliability by reducing the hops between the mesh nodes and the Ryu controller. The indoor testbed is tested before testing the outdoor testbed which is the future work to check whether the routing patterns of the network, the rerouting scenarios after the failure of the primary routes, and the road traffic monitoring application applied in the testbed are able to work correctly or not. According to the results obtained from the SDWMN in-band testbed, the routing and rerouting application is confirmed to function properly. For the data plane testing, three sub-testings are tested. The testing results of indoor SDWMN in-band discussed in this paper shows that the system can function properly and becomes ready for future testing in the real outdoor environment.

Acknowledgment

This work was supported by the Asi@Connect's Data-Centric IoT-Cloud Service Platform for Smart Communities (IoTcloudServe@tEIN) project with grant contract ACA 2016/376-562 under the umbrella of Smart-Mobility@Chula demonstration site.

References

[1] T. N. TomTom N. V. Dutch Company, Amsterdam, “Tomtom traffic index: Bangkok, Thailand” https://www.tomtom.com/engb/trafficindex/city/bangkok. ccessed: 2020, May.

[2] Wi-Fi. https://www.wi-fi.org/. Accessed: 2020, May

[3] LoRaWan. https://lora-alliance.org/. Accessed: 2020 May.

[4] ZigBee. https://www.zigbee.org/. Accessed: 2020 May.

[5] WiMAX. http://wimaxforum.org/. Accessed: 2020 May.

[6] M. Seyedzadegan, M. Othman, B. M. Ali, and S. K. Subramaniam, “Wireless Mesh Networks:WMN Overview, WMN Architecture,” in International Conference on Communication Engineeringand Networks IPCSIT, vol. 19, pp. 12–18, 2011.

[7] M. E. M. Campista, P. M. Esposito, I. M. Moraes, L. H. M. K. Costa, O. C. M. B. Duarte, D. G.Passos, C. V. N. De Albuquerque, D. C. M. Saade, and M. G. Rubinstein, “Routing metrics andprotocols for wireless mesh networks”, IEEE Network, vol. 22, no. 1, pp. 6–12, 2008.

[8] Open Network Foundation, “Software-defined networking: The new norm for networks,” Tech. Rep., Open Networking Foundation, April 2012.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, andJ. Turner, “Openflow: Enabling innovation in campus networks,”Computer Communication Review,vol. 38, pp. 69–74, 04 2008.

[10] A. Jalili, H. Nazari, S. Namvarasl, and M. Keshtgari, “A comprehensive analysis on control planedeployment in sdn: In-band versus out-of-band solutions,” pp. 1025–1031, 12 2017

[11] S. Y. Htet, “Design and implementation of medium-range outdoor wireless meshnetwork with open-flow in raspberry pi,” Master’s thesis, Chulalongkorn University, Bangkok, Thailand, 2018.

[12] A. M. Htut, “Development of near real-time wireless image streaming cloud with apache kafka forroad traffic monitoring application,” Master’s thesis, Chulalongkorn University, Bangkok, Thailand, 2019.

[13] P. Dely, A. Kassler, and N. Bayer, “Openflow for wireless mesh networks,” in 2011 Proceedingsof 20th International Conference on Computer Communications and Networks (ICCCN), pp. 1–6, 2011.

[14] A. Detti, C. Pisa, S. Salsano, and N. Blefari-Melazzi, “Wireless mesh software defined networks(wmsdn),” in2013 IEEE 9th International Conference on Wireless and Mobile Computing, Net-working and Communications (WiMob), pp. 89–95, 2013.

[15] Won Jin Lee, Jung Wan Shin, Hwi Young Lee, and Min Young Chung, “Testbed implementationfor routing wlan traffic in software defined wireless mesh network,” in 2016 Eighth InternationalConference on Ubiquitous and Future Networks (ICUFN), pp. 1052–1055, 2016.

[16] H. Yang, B. Chen, and P. Fu, "Openflow-based load balancing for wireless mesh network," pp. 368–379, 01 2015.

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

[18] K. P. Arun, A. Chakraborty, and B. S. Manoj, "Communication overhead of an openflow wirelessmesh network," in 2014 IEEE International Conference on Advanced Networks and Telecommunca-tions Systems (ANTS), pp. 1–6, 2014.

[19] A. V. Mamidi, S. Babu, and B. S. Manoj, "Dynamic multi-hop switch hand-offs in software de-fined wireless mesh networks," in 2015 IEEE International Conference on Advanced Networks andTelecommuncations Systems (ANTS), pp. 1–6, 2015.

[20] K. Bao, J. D. Matyjas, F. Hu, and S. Kumar, "Intelligent software-defined mesh networks with link-failure adaptive traffic balancing," IEEE Transactions on Cognitive Communications and Networking, vol. 4, no. 2, pp. 266–276, 2018.

[21] R. K. Sriramulu, "Constructing dynamic ad-hoc emergency networks using software-defined wireless mesh networks," Master's thesis, San Jose State University, San Jose, CA, USA, 2018.

[22] C. Yu, Z. Yang, X. Chen, and J. Yang, "Scalable video transmission in software defined wirelessmesh network," in 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft),pp. 456–461, 2018.

[23] Raspberry Pi 3. https://www.raspberrypi.org/. Accessed: 2020 May.

[24] Intel®NUC 7i7BNH.
https://www.intel.com/content/www/us/en/products/boards-kits/nuc/kits/nuc7i7bnh.html. Accessed: 2019 January.

[25] EDUP EP-AC1605. http://www.szedup.com/. Accessed: 2020 May.

[26] Open vSwitch. http://openvswitch.org. Accessed: 2020 May.

[27] RYU SDN Framework. https://osrg.github.io/ryu-book/en/Ryubook.pdf. Accessed: 2020 May.

[28] Ubuntu-Mate. https://ubuntu-mate.org/. Accessed: 2020 May.

[29] Ubuntu. https://www.ubuntu.com/. Accessed: 2020 May