Proceedings of the APAN – Research Workshop 2019
ISBN 978-4-9905448-9-8

SDN-based Coordination for IoT-Cloud Connectivity employing Wired/Wireless Multi-Access SmartX Boxes

Juseong Kim, Jun-Sik Shin and JongWon Kim

Abstract— **Diversified Internet of Things (IoT) -related services typically require networking to the cloud/edge-cloud resources to process and store data from distributed IoT device boxes. In addition, various IoT-related services encourage leveraging different access networking, so IoT device boxes having multiple interfaces are becoming typical configuration. In order to efficiently provide IoT-cloud connectivity via multiple interfaces, multi-access networking is becoming a popular research keyword. And supporting reliable data transmission of IoT data to the cloud is an important feature of multi-access networking. In this paper, to cope with the emerging multi-access networking, we propose SmartX miniBox and SDN-based coordination functionality. SmartX miniBox is a physical box designed to support multi-access networking with SDN-enabled wired interface and OVS-integrated WiFi interfaces. And SDN-based Coordination functionality coordinates SmartX miniBox and IoT device boxes in order to enhance reliability in data transmission. The coordination includes alternating access interfaces in IoT devices boxes and changing networking paths in multi-path wired topology when networking failures occur.**

Index Terms— **Internet of things (IoT), software defined networking (SDN), multi-access, multi-path control, SmartX miniBox**

I. Introduction

With the growing popularity of IoT (Internet of things), the types of IoT device boxes and their network interfaces are diversified. Gartner estimates that up to 20 billion devices will be deployed by 2020 [1]. In addition, the overall amount of data from IoT device boxes is also rapidly growing as the number of IoT device boxes increases, so various IoT services demand computing and storage resources for processing such huge IoT data. Therefore, reliable data transfer from IoT device boxes to cloud/edge-clouds as well as supporting diversified networking is a very interesting research topic in the IoT domain.

Under the emerging multi-access networking, we attempt to prepare an edge-cloud testbed for supporting IoT-cloud service realization. As a preliminary step, we implement SmartX miniBox which can support Ethernet-based wired and WiFi-based wireless networking for IoT device boxes to be able to transfer data to edge-cloud. However, realizing the SmartX miniBox with these interfaces is difficult. For this reason, Software-Defined Networking (SDN) is used to control data transmission and access networking [2]. By using SDN, it can choose the interface to transmit data within various interfaces of the IoT device box. It can also provide reliable networking to recover to network failures during data transmission.

In this paper, we introduce SmartX miniBox, which is supporting wired/WiFi controlled by ONOS (Open Networking Operating System) SDN controller. SmartX miniBox is a prototype version for a single box supporting multi-access networking [3]. In addition, we introduce the SDN-based coordination functionality that controls the transmission of data in a multi-access and multi-path networking environment using SmartX miniBox. The SDN-based coordination functionality helps to reliable and persistent data collection from the IoT device box to the SmartX miniBox. This work is based on a flow steering supporting SDN-based flow control research [4]. The flow steering functionality is extended to support WiFi access.

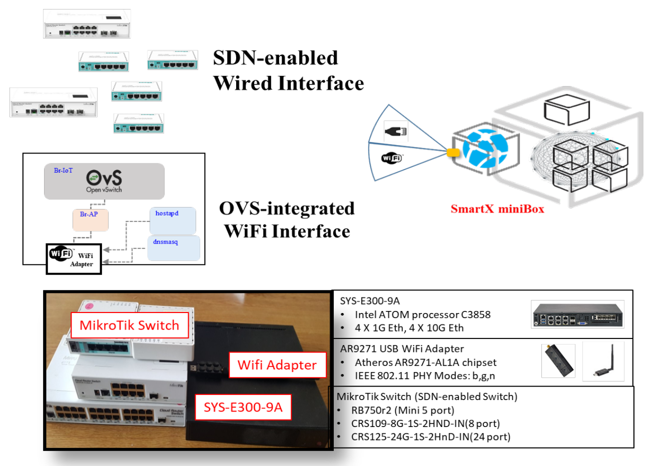
II. SmartX miniBox with Wired/Wireless Multi-Access

SmartX Box is a hyper-converged box that can provide compute, networking and storage resources together [5]. In addition, the SmartX box is implemented as commodity hardware with Linux-based open source software to efficiently

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2015-0-00575, Global SDN/NFV OpenSource Software Core Module/Function Development). This work is also partially supported by the Data-centric IoT-cloud service platform for smart communities (IoTcloudServe@TEIN) project under the WP4 Future Internet of Asi@Connect.

Juseong Kim, Jun-Sik Shin and JongWon Kim are with the school of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), 123 Cheomdangwagi-ro, Buk-gu, Gwangju, 61005, Republic of Korea (e-mail: {jskim, jsshin, jongwon}@nm.gist.ac.kr).

utilize the hyper-converged resources. Based on the SmartX box concept, we attempt to prototyping SmartX miniBox that can partially provide SDN-based multi-access networking to IoT device boxes in order to support IoT-cloud service realization. We design a basic concept and hardware specification for a preliminary prototype of SmartX miniBox as shown in Fig. We select the SDN-enabled wired interface and the OVS-integrated WiFi interface for the preliminary prototype's interface options since these interfaces are widely used in many use cases.



SYS-E300-9A
- Intel ATOM processor C3858
- 4 X 1G Eth, 4 X 10G Eth

AR9271 USB WiFi Adapter
- Atheros AR9271-AL1A chipset
- IEEE 802.11 PHY Modes: b,g,n

MikroTik Switch (SDN-enabled Switch)
- RB750r2 (Mini 5 port)
- CRS109-8G-1S-2HND-IN(8 port)
- CRS125-24G-1S-2HnD-IN(24 port)

Fig. 1. SmartX miniBox Interface Options & Hardware Configuration

The first interface is an SDN-enabled wired interface that is directly connected to SDN-enabled multi-path topology. The multi-path topology consists of several SDN-enabled wired switches. The wired interface of the SmartX miniBox connected to the multi-path topology is connected to the internal SDN-enabled Open vSwitch(OVS)-bridge so that the wired interface can be controlled by the SDN controller. So it can provide multiple available paths between IoT device boxes and the SmartX miniBox under the control of the SDN controller. Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license [6]. OVS can directly support OpenFlow. This interface provides networking stability by providing an alternate path for dynamically transmitting IoT data in the event of a networking failure under the control of the SDN controller. We utilize an open-source SDN controller named ONOS (Open Networking Operating System) [7]. To configure the multi-path topology for a wired interface, we use four SDN-enabled switches that are two types of MikroTik switches (CRS109-8G-1s-2HnD-IN [8], and RB750R2 [9]) supporting OpenFlow 1.0. They can be easily configured using a GUI tool named WinBox [10].

The second interface is an OVS-integrated WiFi interface that can support wireless data transmission controlled by an SDN controller. WiFi networking is one of the most popular wireless networking technology, so we decide to put the WiFi interface into the first prototype of SmartX miniBox. The internal network configuration for the OVS-integrated WiFi interface is shown in Fig. 2. But, the WiFi interface itself is hard to be controlled by an SDN controller. So we connect the WiFi interface to the SDN-enabled OVS-bridge used to configure the SDN-enabled wired interface. So, this configuration allows the WiFi network can be controlled by an SDN controller. To provide AP functionality, we used an open source AP daemon hostapd [11] and used a dnsmasq for the DHCP server [12].

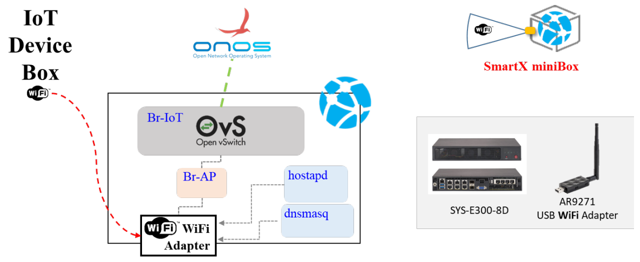


Fig. 2. OVS-integrated WiFi interface internal Network Configuration

III. SDN-Based Coordination Functionality For IoT Devices

We define SDN-based coordination as a software solution that can support reliable data transfer from IoT device boxes to SmartX miniBox by using the SDN controller. Fig. 3 shows the proposed coordination concept and a testbed configuration where we apply the coordination. IoT device boxes (left side in Fig. 3) can reach SmartX miniBox (right side in Fig. 3) through a multi-path wired network and WiFi-based wireless network. All networking interfaces in the IoT device box are directly attached to SDN-enabled OVS-bridge. The configuration makes IoT device boxes act as a single SDN-enabled switch. At the top in Fig. 3, the coordination software with the SDN controller can monitor/control networking interfaces in IoT device boxes and SmartX miniBox.

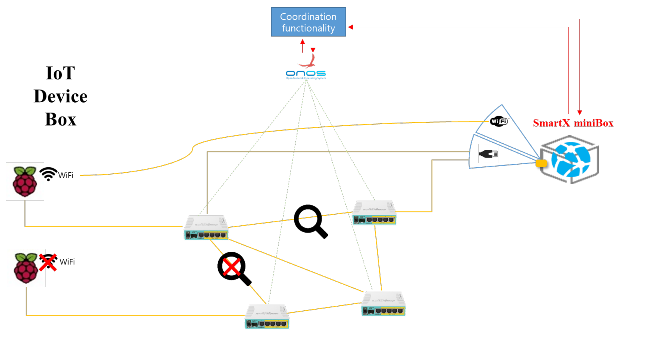


Fig. 3. A concept of SDN-based Coordination for IoT connectivity

For feature verification, we attach Docker containers [13] on the configured OVS-bridge in SmartX miniBox and IoT device boxes. In order to simplify coordination scenarios, we assume all containers are working on the same network, thus we did not consider L3 routing in this paper. So, all the

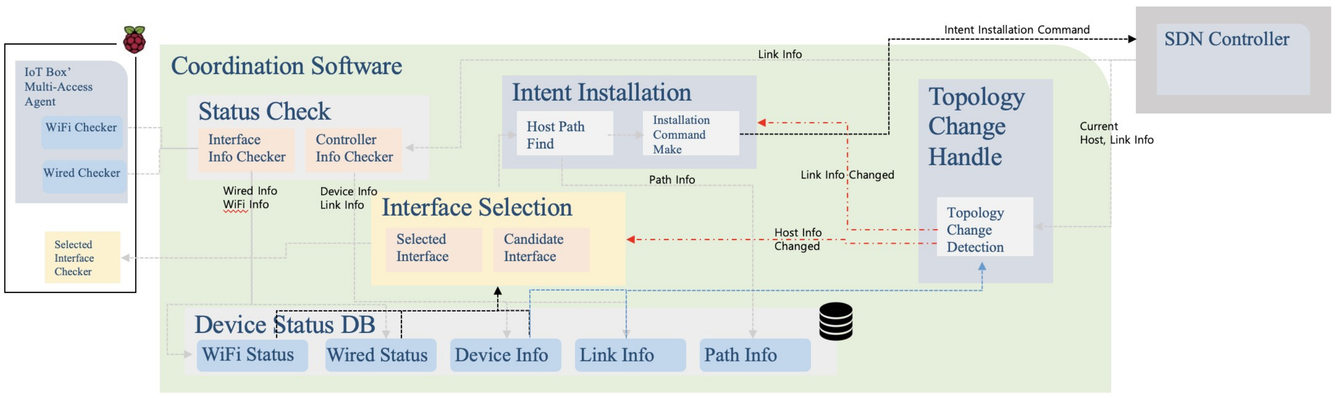


Fig. 4. Software design of Multi-Access Coordination functionality

containers have IP addresses in the same subnet, and the SDN-based coordination functionality controls communication between containers.

We consider the functional requirements of SDN-based coordination functionality in order to support reliable data transfer. First, the coordination functionality may change currently used interface to another (e.g., from OVS-integrated WiFi interface to SDN-enabled wired interface) for data transfer, if the current interface becomes unavailable due to, for example, link failure. When IoT device box use wired network for data transfer, the coordination functionality should dynamically select an available path from multiple possible paths even if the current path occurs failures. For reliable WiFi networking, the coordination software can change the WiFi channel if the networking speed becomes slow due to networking congestion. In addition, SDN-based coordination functionality may support reliable data transfer between the IoT device box and SmartX miniBox. For example, if an IoT device box sends 100 packets, the coordination functionality may ensure all these packets have successfully arrived at SmartX miniBox.

To satisfy the requirements, we design SDN-based coordination functionality as shown in Fig. 4.

- **Status Check:** Check the status of interface and switch by comparing the data received from IoT box's multi-access agent with the information received from ONOS RESTful API. And stores data from ONOS RESTful API data to the Device Status DB.
- **Device Status DB:** Database that stores the result of the check of Status Check. It stores 5 types of data. WiFi status indicates the WiFi interface status of IoT device boxes. Wires Status indicates wired interfaces status of IoT device boxes. Device Info indicates all the SDN-enabled Switch's status (SDN-enabled switch on the multi-path topology and SDN-enabled OVS-bridge in the IoT device boxes and the SmartX miniBox.), Link Info indicates all the network link on the multi-path topology. And Path Info indicates the datapath which is chosen to send data through on the multi-path topology.
- **Interface Selection:** Choose the interface to transmit data of the IoT device box based on the data in Device Status DB. The selected interface information is stored in the Selected Interface Info table and the others are stored in the Candidate interface Info table.
- **Intent Installation:** Find the datapath on the multi-path topology on the help of the SDN controller. And then, specify flow rules to enable transfer data through the selected interface and path on the multi-path topology. Use the ONOS intent framework to specify the flow rule.
- **Topology Change Handle:** Detects network failures. When the interface of IoT device boxes, Interface Selection is performed to find another interface to transmit data. When datapath is disconnected, the Intent Installation process is performed to choose the alternative path from IoT device boxes to SmartX miniBox and specify the new flow rule.

Notice that IoT device boxes have a multi-access agent that collects and sends the status of networking interfaces to the SDN-based coordination functionality, however, the agent is not included in the prototype implementation. Thus we create a JSON file that contains interfaces list and the SDN-based coordination functionality may collect the JSON files in IoT device boxes to obtain interfaces list and status.

In order for the SDN-based coordination functionality to work, the SDN controller must be able to check the connection status of all of the IoT device boxes and SmartX miniBox. The SDN-based coordination functionality calls the RESTful API from the SDN controller to check the connection status. The checklist is the status and connection structure of the switches in the multi-path topology, the interface list and the connection status of the IoT device box. Base on checklist information, the SDN-based coordination functionality performs path and interface control through three steps: Initialization, Interface & Path Selection, and Interface & Path Transfer.

The Initialization step is performed when the SDN-based coordination functionality is started. It is designed to receive the information of the MAC address and state of the interface from the multi-access agent operating in the IoT device box. This MAC address is compared with the device information (SDN-enabled Switch information) from the RESTful API of the ONOS SDN controller. When the comparison is complete, SDN-based coordination functionality can classify which device is OVS-bridge in the IoT device box. After the comparison, the wired/wireless interface state of the IoT device box, the MAC address of each interface, and switch information on the multi-path are stored in the Device Status DB. Also, the link information existing on the multi-path is stored in the Device Status DB. The Link information will be used in the Interface & Path selection step.

At the Interface & Path selection step, select the interface of the IoT device box that supports wired/wireless interfaces at the same time. When the wire is selected, SDN-based coordination functionality chooses the path to transmit data on the multi-path topology. When selecting the interface, the software takes the interface information from Device Status DB and selects the interface. The selected interface information is stored in the Selected Interface Info table in the Interface Selection, and the unselected interface is stored in the Candidate Interface Info table in the Interface Selection. The datapath between the IoT Device boxes and SmartX miniBox is calculated by the SDN controller. At this time, the SDN controller calculates the shortest path including the Selected Interface Info. After selecting the path, the flow rule is specified by using the Point-to-Point Intent of the ONOS Intent framework for the path found. Then, the path information that specified the flow rule is stored in the Device Status DB as a Path Info. When installing a Point-to-Point Intent, we record the MAC address of source and destination. By recording the MAC address, we can specify multiple flow rules on a single port.

The last step is the Interface & Path transfer step. SDN-based coordination functionality checks the network failure once every 30 seconds. The SDN-based coordination functionality uses the RESTful API of the SDN controller to retrieve the link information of the current state at the time of determining the failure. Check whether the Selected Interface Info information is stored in this link information. If there is no corresponding information, it is determined that the connection of the interface is disconnected. And the Interface & Path Selection process is performed again using the Candidate Interface Info to transfer interface. In the case of path transfer, the link information of the current point is compared with the Path Info in the Device Status DB. If the path of the Path Info is not in the Link information, it recognizes that the datapath is disconnected. Then, a flow rule is specified to the new path through the interface & path selection process.

IV. SDN-BASED COORDINATION FUNCTIONALITY VERIFICATION

In this section, we discuss the verification of the SDN-based coordination functionality. We verify the functionality using some scenarios in the environment for validation. We prepare two IoT device boxes. One supports both wired/wireless interface at the same time. The other one supports only the wired interface. And we connect additional networks to IoT device boxes to receive control signals from the SDN controller. The testbed is configured like Fig. 5. On the bottom, the blue boxed Raspberry Pi uses both wired/wireless interfaces to transmit data. The Raspberry Pi covered by the red box is an IoT device box that uses only the wired interface. The switch on the far right is the SmartX miniBox.

Four experiments are conducted in this environment. First, we experiment to see if the coordination functionality selects the appropriate interface and path in the initial state to specify the flow rule. Second, we experiment to see the interface transfer when the interface being is to transmit data is disconnected. Third, we verify whether SDN-based coordination functionality can find an alternative path and specify new flow rules when the data path is disconnected. Finally, we send the actual packet and see how quickly the SDN-based coordination functionality can handle network failures.

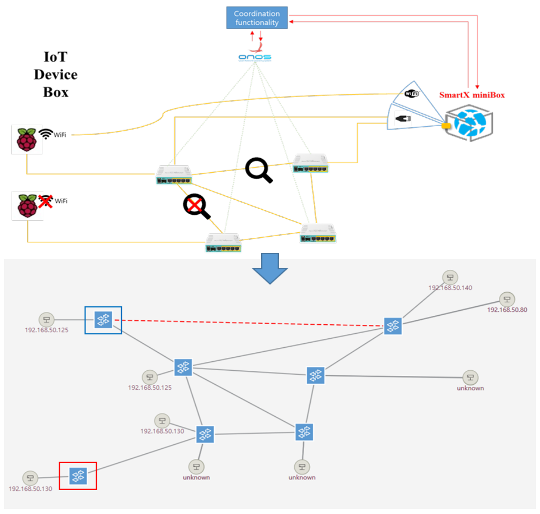


Fig. 5. Testbed for verification

First, we look at the results of specifying the initial flow rule with the help of the SDN-based coordination functionality. The results are shown in Fig. 6. In the case of the IoT device box connected to both wired/wireless interfaces, it can be confirmed that the wireless interface is selected. In the case of an IoT device box using only a wire interface, the result of selecting a path for transmitting data in the multi-path topology can be confirmed.

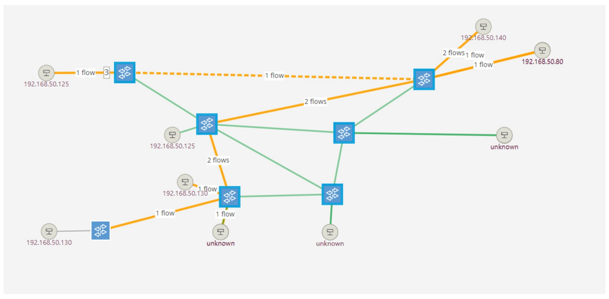


Fig. 6. Result of Interface & Path Selection

The second is an experiment that verifies the interface transfer by disconnecting the WiFi interface of the IoT device box supporting multiple interfaces. The result of the experiment is shown in the right side of Fig. 7. When the wireless connection is disconnected, the SDN-based coordination functionality recognizes it and specify the new flow rules to another interface. In addition, it confirmed that find a new datapath on the multi-path topology to communicate using wired interface.

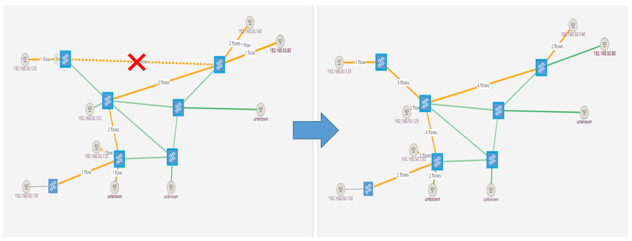


Fig. 7. Result of Interface transfer

Third, we verify that finding an alternative path when the datapath is disconnected. The result is shown in Fig. 8. When the connection of the datapath is disconnected, SDN-based coordination functionality notices the change of the link and finds the alternative path. And then specify new flow rules for that path to communicate through that path.

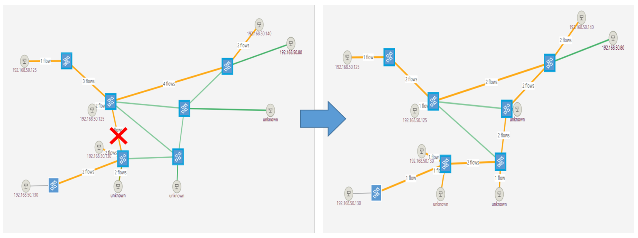


Fig. 8. Result of Path transfer

Next, we discuss the sending actual packet. This experiment confirms transmission. We used the python "scapy" library to generate the packets for verification. It generates and sends 30 packets numbered from 1 to 30 [14]. And it sends the next packet when the echo arrives from the destination. This program only creates and sends packets under the control of SDN-based coordination functionality. When a network failure occurs, the coordination functionality sends a multicast signal to the IoT diversity boxes after recovering the failure. When IoT device boxes receive the signal, IoT device boxes resend packets that they did not receive echo. The result of the sending packets is shown in Fig. 9. The graph is the time for sending each packet and receiving an echo. The network failure occurred during the processing of packets No.10 and No.19. However, the time to recover from a network failure is different. The reason is that multi-access coordination functionality judge network failure in once every 30 seconds. In the case of the No.19 packet, it was time to start the transmission and judge the network failure soon. Because of this reason, the No. 19 packet case was recovered faster than the No. 10 packet case.

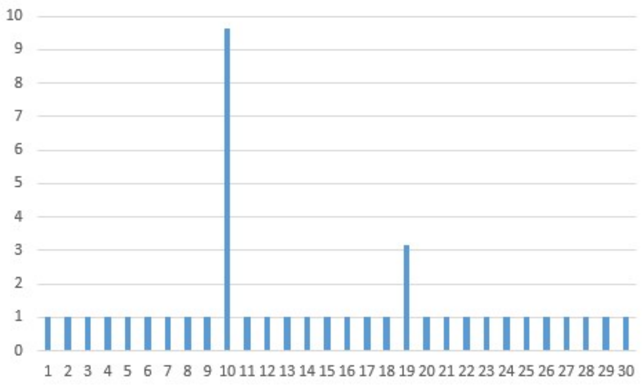


Fig. 9. Packet send/receive time

V. Conclusion

In this paper, we have discussed SmartX miniBox, a single box that supports multiple interfaces: SDN-supported wired interface, and OVS-integrated WiFi interface. We also discussed SDN-based coordination functionality that enables SmartX miniBox to collect IoT data persistently from the IoT device box. In addition, we verified the main functionality on a small testbed which is built to verify the SDN-based functionality. Finally, we verified the reliable data transmission on the control of SDN-based coordination functionality by sending actual packets.

References

[1] Gartner. (2017, February). Gartner says 8.4 billion connected "Things" will be in use in 2017, up 31 percent from 2016. [Online]. Available: https://www.gartner.com/newsroom/id/3598917.

[2] C. S. Li and W. Liao, "Software defined networks," *IEEE Communications Magazine*, Vol. 51, No. 2, pp. 113-113, Feb. 2013.

[3] J. Kim, et al. "Preparing SDN-based Path Diversity Coordination for Multi-Access Edge Cloud," *in Proc. KICS 2017 Fall Conference*, Daegu, Republic of Korea, Nov. 2017.

[4] H. Yun, et al., "Design and Verification of SmartX IoT-Cloud Hub employing SDN-assisted Flow Steering," *KIISE Transactions on Computing Practices*, Vol. 24, No. 10, pp. 493-504, Oct. 2018.

[5] S. Lee, et al., "Reconfiguration of OF@KOREN Playground Infrastructure for Converged SmartX Playground," *in Proc. KICS 2018 Fall Conference*, Seoul, Republic of Korea, Nov. 2018.

[6] OpenvSwitch. [Online]. Available: https://www.openvswitch.org/

[7] P. Berde, et al., "ONOS: towards an open, distributed SDN OS," *in Proc. HotSDN 14*, Aug. 2014.

[8] MikroTik Cloud Router Switch Official data sheet. [Online]. Available: https://i.mt.lv/cdn/rb_files/CRS109-8G-1S-2HnD-IN-150714082814.pdf

[9] MikroTik RouterBOARD 750/GL official data sheet. [Online]. Available: https://i.mt.lv/cdn/rb_files/rb750gl-ug.pdf

[10] MikroTik Winbox manual. [Online]. Available: https://wiki.mikrotik.com/wiki/Manual:Winbox

[11] J. Malinen. (2013, Jan.). Hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator. [Online]. Available: https://w1.fi/hostapd/

[12] S. Kelley. (2008). Dnsmasq. [Online]. Available: http://www.thekelleys.org.uk/dnsmasq/doc.html

[13] Using OVS bridge for docker networking. [Online]. Available: https://developer.ibm.com/recipes/tutorials/using-ovs-bridge-for-docker-networking/

[14] thePacketGeek (2013). Sending and Receiving with Scapy. [Online]. Available: https://thepacketgeek.com/scapy-p-06-sending-and-receiving-with-scapy/