# High Availability WLANs Based on Software-Defined Networking

Hwi Young Lee, Young Min Kwon, Jung Wan Shin, Won Jin Lee, and Min Young Chung

College of Information and Communication Engineering, Sungkyunkwan University,
Email: {lhy152, ko116, withmyjw, reign0208, mychung}@skku.edu

***Abstract*—The software defined networking (SDN) has become one of the popular subject in the domain of information and communication technology and a huge amount of research has been conducted in this area. However, most of these existing research works only provide theoretical SDN concepts and they hardly show any implementation or testbed results. In order to market the SDN technology, useful real-time practical applications of SDN controller are required. Therefore, in this paper we propose a novel SDN based high availability (HA) solution for wireless local area networks (WLAN). The proposed solution uses open network operating system (ONOS) controller as a SDN controller. Our experimental results under real-time network environment show that the ONOS controller can be used to manage the overall WLANs.**

***Index Terms*—Software defined networking; High availability WLAN; Open network operating system.**

## I. Introduction

Recently, the software defined networking (SDN) has attracted a lot of interest. The main characteristics of SDN is to decouple the control and data planes of a network and provide the freedom of programmability to development more efficient network applications [1]. Furthermore, the SDN also results in less complex and more flexible wired networks [2]. Due to these characteristics, many research works consider to employ SDN architecture to wireless networks such as mobile networks and wireless local area networks (WLANs) [3]-[7].

The existing studies in this area of wireless networks mainly focus on the theoretical SDN concepts of wireless networks and they rarely provide any real-time SDN based results. It is due to the fact that the implementation of real-time SDN controller is very difficult. Furthermore, the existing research works hardly discuss any useful application of SDN controllers. However, in order to market SDN technology the industry requires valuable practical SDN applications. Therefore, in this paper we study the feasibility of SDN controller by implementing a real-time WLAN. As shown in Figure 1, our proposed high availability (HA) scheme aims to provide HA solution for WLAN by using open network operating system (ONOS) controller . In our proposed scheme. When the ONOS controller detects that a WLAN AP is unavailable due to some unexpected system failure, it instructs its neighboring WLAN APs to accommodate the abandoned stations (STAs) which were previously associated with malfunctioned AP. In addition, the ONOS controller also instructs the OpenFlow switches to update their flow tables for efficient and reliable data transfer. As a result, the STAs is not severely affected by the failure of the WLAN AP. The rest of the paper is organized as follows. Section II presents the preliminaries regarding our proposed scheme. In Section III, we introduce the system architecture and procedures for the proposed HA WLANs. Section IV contains the detailed performance evaluation of the proposed HA WLAN and Section V provides the conclusion of this paper.

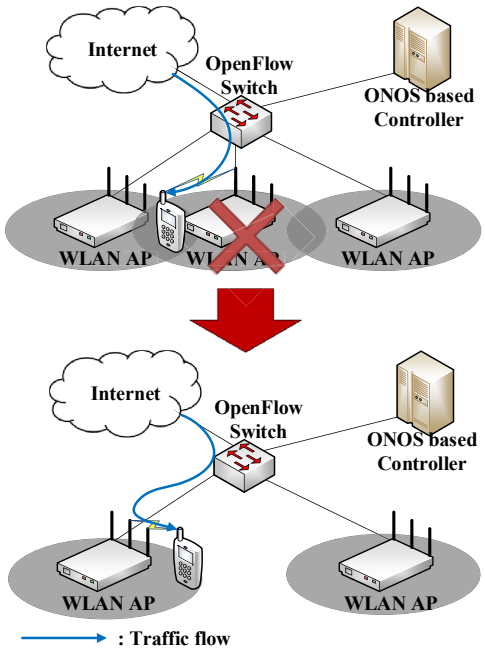



Fig. 1. Solution for High Availability WLANs

## II. Preliminaries

For better understanding of our proposed scheme, this section provides a brief introduction of both OpenFlow and ONOS controller.

### A. OpenFlow

OpenFlow is a standard protocol to provide interface between the control and forwarding layers of SDN architecture [8]. OpenFlow protocol identifies the common features in the flow tables of the Ethernet switches [9] and it manages the flow table of the switch. It enable the user to control switches without any technical support from vendors [10]. The main objective of OpenFlow is to provide a platform

 ISBN: 978-1-61208-473-2 

to test the newly developed networking ideas [9]. Due to these advantages several vendors around the globe are taking keen interest in the standardization procedure of OpenFlow protocol. The OpenFlow standardization procedure is carried out by Open Networking Foundation (ONF) founded in 2011 [10].

OpenFlow network has centralized characteristic where a single controller can manage multiple switches. In OpenFlow network, single controller can analyze traffic statistics and control the traffic flows. OpenFlow network consists of three components: an OpenFlow switch, a OpenFlow channel, and a controller [11]. An OpenFlow switch is comprised of one or more flow tables and a group table. Each flow table of the OpenFlow switch has a set of flow entities including match fields, counters, and instructions for traffic packets. A group table contains group entities with action buckets dependent on group types. The action indicates the additional processing and forwarding features such as multi-path, fast rerouting, and link aggregation. Based on these flow tables and a group table, the OpenFlow switch examines and forwards data packets. An OpenFlow channel is an interface for the OpenFlow switch to communicate with an external controller. Through the OpenFlow channel, the OpenFlow switch can receive the control message to add, update, and delete its flow tables from the controller.

### B. Open Network Operating System (ONOS)

During the past several years, open source SDN controllers such as NOX [12], Beacon [13], and POX [14] had been developed. The objective of these controllers is to explore and demonstrate SDN potential. Since these controllers have primitive programming and devices-oriented abstractions, SDN applications for these controllers are tightly coupled to OpenFlow protocol such as network device drivers. Therefore, these controllers are hard to provide the key features such as scalability and HA.

In order to provide these key features, ONOS is developed [15]. Since ONOS is a network operating system, it is responsible for the following functions: management for finite resources on behalf of resource consumers, isolation and protection of ONOS users from each other, efficient resource management, and security from the outside world. The architecture of ONOS consists of distributed core, Northbound abstraction/APIs, Southbound abstraction/APIs, and software modularity.

- **Distributed core**: It is required to provide the scalability, and HA of the SDN control plane. ONOS can be deployed as a service on a cluster of servers running the same ONOS software. If system failure occurs in an ONOS server, the distributed core enables rapid failover. In addition, a cluster of multiple ONOS servers can perform applications and control network devices from a single platform. This feature makes ONOS more scalable.
- **Northbound abstraction/APIs**: ONOS enables users to easily develop application with the help of Northbound abstraction/APIs. ONOS includes network graph and application intents to ease development of control, management, and configuration services. There are two powerful Northbound abstractions; intent framework and global network view. The intent framework allows an application to request a service from the network without information of how the service will be performed. And, the global network view provides the application a view of the whole network.
- **Southbound abstraction/APIs**: The Southbound abstractions provides the interfaces between OpenFlow control plan and network devices. The Southbound abstraction enables ONOS to control or manage multiple diverse devices, even if they use different protocols such as OpenFlow, NetConf, etc.
- **Software modularity**: ONOS provides the freedom to develop, debug, maintain, and upgrade ONOS as a software system. Because of the modularity, new applications or new protocol adapters can be added according to user requirement. In addition, software modularity provides a architectural integrity and coherence, easy maintenance with fewer side effects of changes, and extensibility and customization of components.

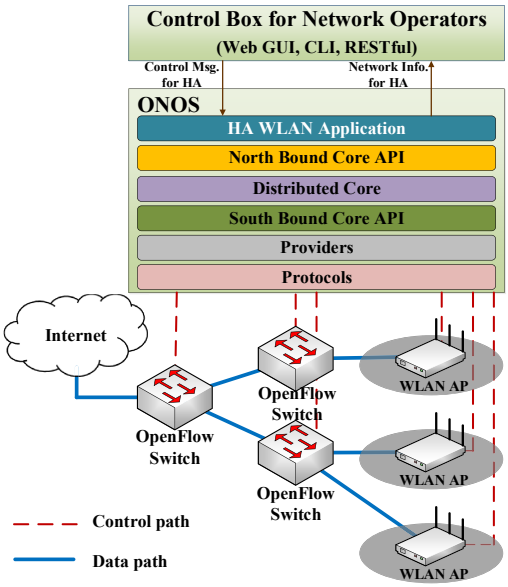



Fig. 2. System Architecture

## III. Proposed Scheme

In this Section, we introduce a SDN-based HA WLAN solution. For implementation of HA WLAN, we propose the system architecture, the monitoring procedure for WLAN APs, and the flow control and failover procedure.

### A. System Architecture

Figure 2 shows the proposed system architecture for supporting the HA WLAN application based on the ONOS

 ISBN: 978-1-61208-473-2 

controller. The system architecture consists of three components; control box for network operators, ONOS controller, and OpenFlow-based network. The control box provides the abstractive view of OpenFlow-based network. Network operators can check the status of OpenFlow-based network and control the OpenFlow-based network via Web graphic user interfaces (GUI), command line interface (CLI), or RESTful of the control box.

The ONOS controller performs the procedures for supporting HA WLANs. It periodically monitors the state of WLAN APs. If the ONOS controller detects the system failure in one or more WLAN APs, it instructs the available WLAN APs to support the re-association of STAs which are abandoned by the dead AP. When the STAs are re-associated to available WLAN APs, the ONOS controller directs OpenFlow switches to update their flow table in order to forward the respective data traffic to newly associated STAs.

The OpenFlow-based network is comprised of several OpenFlow switches and WLAN APs. Since the OpenFlow switches support the OpenFlow protocol, they can be managed by ONOS controller. WLAN APs are entities which provide wireless accesses to their associated STAs. For implementation of HA WLANs, WLAN APs also have interface to receive the control messages from ONOS controller and report their state to the controller.

### B. Monitoring Procedure for WLAN APs

In order to provide HA WLAN, the ONOS controller should detect the system failure of one or more WLAN APs. It is impossible for the ONOS controller to perceive the exact time when the system failure occurs in a WLAN AP. Hence, the ONOS controller should periodically confirm the state of WLAN APs as shown in Figure 3.

The proposed monitoring procedure for state of WLAN APs is as follows: In Step 1, ONOS controller sends *hello* messages to WLAN APs, and then it waits for the response for a predefined time in Step 2. Based on the type of response, the ONOS controller confirms the state of the WLAN APs. As shown in Step 3-a) of Figure 3, if the ONOS controller receives the *ok* messages from a WLAN AP before the timer expires it perceives the state of the WLAN AP as the *available* state. If the ONOS controller receives the *WiFi Failure* message from a WLAN AP such as Step 3-b), it recognize the state of the WLAN AP as the *WiFi-disabled* state which indicates that the WLAN AP is running but its WiFi radio does not work. On the other hand, If the ONOS controller does not receive any message from a WLAN AP after the predefined time such as Step 3-c), it perceives that a system failure has occurred and the WLAN AP is dead. Thus, the ONOS controller sets the state of WLAN APs to the *unavailable* state.

Even if some WLAN APs are malfunctioning, the proposed HA WLAN system should provide seamless services to users and in order to do this, ONOS controller determines alternative WLAN APs which can accommodate the abandoned STAs. If

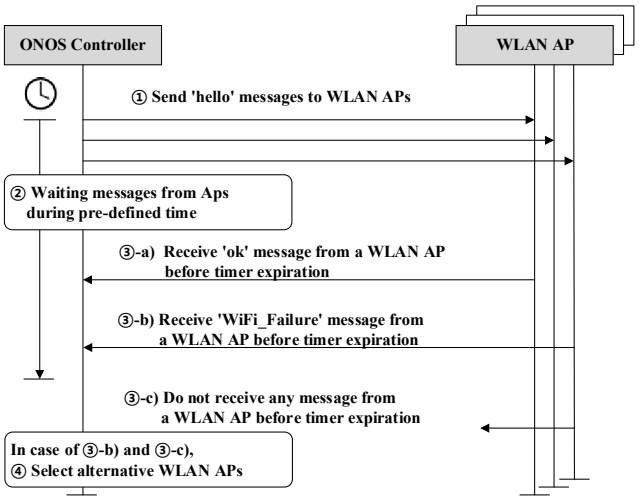



Fig. 3. Procedure to monitor the state of WLAN APs

ONOS controller detects a WLAN AP under the *WiFi-disabled* or *unavailable* (Step 4), it requests the re-association and re-authorization for the abandoned STAs to the newly assigned APs.

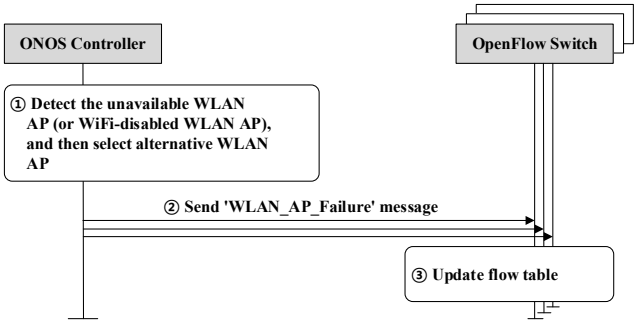



Fig. 4. Procedure to control traffic flow

### C. Flow Control and Failover Procedure

ONOS controller should adjust the traffic path in order to prevent the data from being forwarded to the disabled WLAN AP. If ONOS controller detects a disabled WLAN AP under the *WiFi-disabled* or *unavailable* state, it sends *WiFi AP Failure* messages to OpenFlow switches as shown in Figure 4. A *WiFi AP Failure* message includes IP addresses and port numbers of the disabled WLAN AP and its alternative WLAN AP. Based on the information of the *WiFi AP Failure* message, OpenFlow switches delete their flow tables and stop forwarding data packets to the faulty WLAN AP. Then, they add new flow tables and forward the data packets to the alternative WLAN AP.

If state of a malfunctioned WLAN AP is the *WiFi-disabled*, it means that the WLAN AP is still running with null WiFi signal. In this case, ONOS controller can command the

 ISBN: 978-1-61208-473-2 

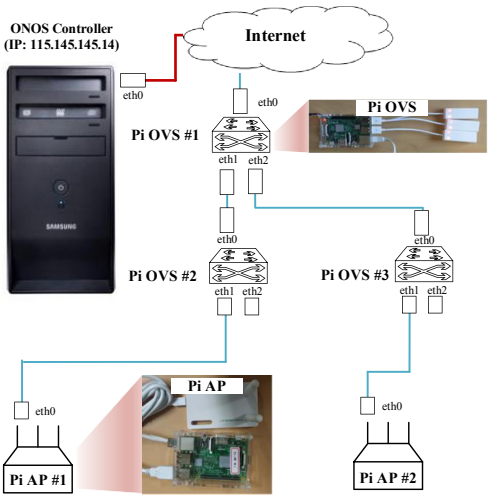
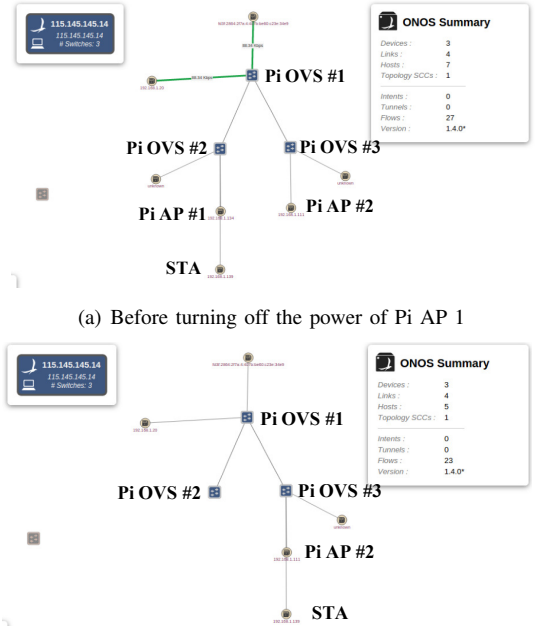



Fig. 5. Network components for test environment




(a) Before turning off the power of Pi AP 1

(b) After turning off the power of Pi AP 1

Fig. 6. Network topology in the Web GUI of ONOS

WLAN AP to reboot. After rebooting, if the WLAN AP successfully recovers, the ONOS controller confirms the state of the WLAN AP by performing the monitoring procedure, and then it changes the state of the WLAN AP from the *WiFi-disabled* state to the *available* state. The ONOS controller requests that the WLAN AP performs the re-association and re-authorization for its STAs. Simultaneously, it sends the *WiFi AP Failover* message with IP addresses and port numbers of the WLAN AP and its alternative WLAN AP. By using *WiFi AP Failover* message, OpenFlow switches update the flow tables of OpenFlow switches in order to forward data packets to the WLAN AP.

## IV. Performance Evaluation

In this Section, a detailed description of our test environment for the performance evaluation of HA WLAN is provided. Under the real-time test environment, we measure and analyze the data rate by using WireShark application [16].

### A. Test Environment

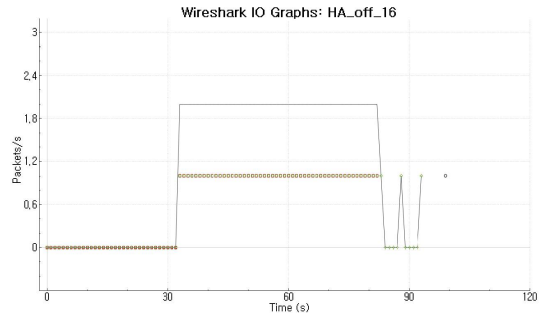
For performance evaluation of HA WLAN, we construct test environment with one ONOS controller, three OpenFlow switches, and two WLAN APs as shown in Figure 5. The ONOS controller is made by installing the ONOS Drake package (version 1.3.0) on a desktop PC. Since the ONOS controller has fixed global IP address, OpenFlow switches can be connected with the ONOS controller via Internet.

For implementation of OpenFlow switch and WLAN AP, Raspberry Pi [17] which is one of open source hardware (OSHW) platforms is utilized. The Raspberry Pi supports the Raspbian OS based on the debian linux [17]. It can perform the functionalities of OpenFlow protocol by installing the Open vSwitch packages [18]. In our test environment, we define the Raspberry Pi using Open vSwitch as Pi OVS. As shown in Figure 5, three Pi OVSs are deployed as a binary tree, where all Pi OVSs manages the packet forwarding based on their flow tables. Two WLAN APs used in our test environment comprise WLAN USB adaptors having MediaTek RT5572 chipsets and Raspberry Pi. The Raspberry Pi can perform the functionalities according to IEEE 802.11 standards by installing the hostapd package [19]. We define the WLAN APs as Pi AP. The Pi APs 1 and 2 are connected with Pi OVSs 2 and 3, respectively. One laptop PC with IEEE 802.11 WLAN radio interface is used as a STA. In our test environment, Pi APs and the STA utilize the 5GHz ISM band to communicate with each other, which is not used by any other WLAN hot spot in the neighborhood.

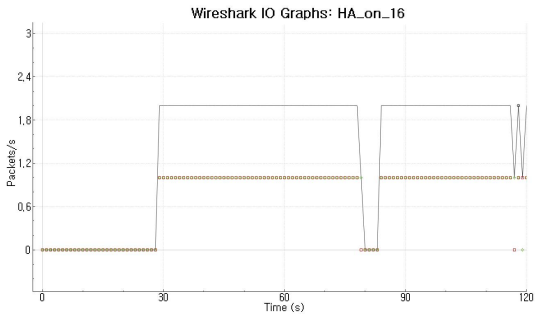
### B. Experimental Result

Under our test environment, we consider a test scenario to confirm feasibility of our proposed HA WLAN solution. In the HA WLAN scenario, we consider that the STA is associated with the Pi AP 1. We configure that the Pi AP 2 is the alternate WLAN AP of the Pi AP 1. In case that the power of a WLAN AP (i.e., PI AP 1) is turned off, we check whether the STA associated to the PI AP 1 is reconnected to the alternative WLAN AP (i.e., Pi AP 2) or not. Since ONOS controller provides the Web GUI showing the network topology. we confirm the execution of the our proposed HA WLAN solution through the Web GUI of ONOS controller as shown in Figure 6. Figure 6(a) shows the initial condition where the Pi AP 1 with an STA is running normally. When the power of Pi AP 1 is cut off (that is, the Pi AP 1 is disabled), an STA will be re-associated to the Pi AP 2 due to HA WLAN solution of ONOS controller. By Figure 6(b), we confirm that the STA is successfully re-associated to the Pi AP 2.

 ISBN: 978-1-61208-473-2




(a) Not employing HA WLAN solution




(b) Employing HA WLAN solution

Fig. 7. Packet data rate

For estimating the effect of HA WLAN, we measure the packet data rate by using WireShark application [16]. Figure 7(a) and 7(b) show that the effect of our proposed scheme for high availability when WLAN AP is disabled. We send packets from domain name system (DNS) server of google IP to STA, then power off Pi AP 1 at 80 seconds. As shown in Figure 7(a), STA can not receive the packets after Pi AP 1 is disabled. It means that other WLAN AP (i.e., PI AP 2) do not recognize the disconnection between Pi AP 1 and STA in the ordinary environment. However, Figure 7(b) presents the successful re-association from Pi AP 1 to the Pi AP 2 with HA WLAN solution of ONOS controller. After being disabled Pi AP 1, Pi AP 2 can associate with STA by using HA WLAN solution of ONOS controller. Thus, after 4 seconds, STA can receive the packet from Pi AP 2 instead of Pi AP 1. Through these experimental results, we confirm the feasibility of our proposed scheme.

## V. Conclusion

In this paper, we introduced an ONOS SDN controller based HA WLAN solution for WLANs. Unlike previous research works where only theoretical concepts of SDN are discussed, we developed a real-time ONOS controller based WLAN test environment. Furthermore, we also implemented and evaluated the performance of our proposed HA WLAN solution. According to our proposed scheme, the SDN controller periodically monitors the state of its attached WLAN APs and in case of an AP failure it re-associates it abandoned STAs to a neighboring functional AP. Through experimental results, we confirm the successful implementation of our proposed HA WLAN scheme. As part of our future work, we plan to reduce the re-association time by enhancing the algorithm and to make the scenario more realistic.

## Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (B0190-15-2013, Development of Access Technology Agnostic Next-Generation Networking Technology for Wired-Wireless Converged Networks)

Prof. Min Young Chung is the corresponding author.

## References

[1] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys & Tutorials*, Vol. 17, no. 1, pp. 27-51, Jun. 2014.

[2] D. Raumer, L. Schwaighofer, and G. Carle, "Monsamp: A distributed SDN application for QoS monitoring," in *Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 961-968, Sep. 2014.

[3] T. Chen, M. Matinmikko, X. Chen, X. Zhou, and P. Ahokangas, "Software defined mobile networks: concept, survey, and research directions," *IEEE Communications Magazine*, Vol. 53, no. 11, pp. 126-133, Nov. 2015.

[4] F. Granelli, et al. "Software defined and virtualized wireless access in future wireless networks: scenarios and standards," *IEEE Communications Magazine*, Vol. 53, no. 6, pp. 26-34, Jun. 2015.

[5] D. Zhao, M. Zhu, and M. Xu, "Supporting One Big AP illusion in enterprise WLAN: An SDN-based solution," *Sixth International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1-6, Oct. 2014.

[6] R. Riggio, M. Marina, J. Schulz Zander, S. Kuklinski, and T. Rasheed, "Programming Abstractions for Software-Defined Wireless Networks," *IEEE Transactions on Network and Service Management*, Vol. 12, no. 2, pp. 146-162, Jun. 2015.

[7] D. Zhao, M. Zhu, and M. Xu, "SDWLAN: A flexible architecture of enterprise WLAN for client-unaware fast AP handoff Computing," *International Conference on Communication and Networking Technologies (ICCCNT)*, pp.1-6, Jul. 2014.

[8] OpenFlow, Accessed on 11 April 2016. [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow

[9] N. McKeown, et al. "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, Vol. 38, no. 2, pp. 69-74, Apr. 2008.

[10] A. Lara, A. Kolasani, and B. Ramamurthy, "Network Innovation using OpenFlow: A Survey," *IEEE Communications Surveys & Tutorials*, Vol. 16, no. 1, pp. 493-512, 2014.

[11] OpenFlow Switch Specification, v1.3.2, 2013.

[12] N. Gude, et al. "Towards opportunistic flow management in OpenFlow," *ACM SIGCOMM Computer Communication Review*, Vol. 38, no. 3, pp. 105-110, July 2008.

[13] What is Beacon? Accessed on 11 April 2016. [Online]. Available: https://openflow.stanford.edu/display/Beacon/Home

[14] POX wiki, Accessed on 11 April 2016. [Online]. Available: https://openflow.stanford.edu/display/ONL/POX+Wiki

[15] ON.LAB White paper, Introducing ONOS - a SDN network operating system for Service Providers, 2014.

[16] WireShark, Accessed on 11 April 2016. [Online]. Available: https://www.wireshark.org/

[17] Raspberry Pi, Accessed on 11 April 2016. [Online]. Available: https://www.raspberrypi.org/

[18] Open vSwitch, Accessed on 11 April 2016. [Online]. Available: http://openvswitch.org/

[19] hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator, Accessed on 11 April 2016. [Online]. Available: https://w1.fi/hostapd/

 ISBN: 978-1-61208-473-2 