An Extended SDN Controller for Handover in Heterogeneous Wireless Network

Toan Nguyen-Duc
Graduate School of Engineering and Science
Shibaura Institute of Technology
Tokyo, Japan
nb14501@shibaura-it.ac.jp

Eiji Kamioka
Graduate School of Engineering and Science
Shibaura Institute of Technology
Tokyo, Japan
kamioka@shibaura-it.ac.jp

Abstract— **In software defined networks, a centralized control is commonly used to provide a global view of the network and to simplify the network operation as well as network programmability. However, in the centralized model, the control path between the network devices and the controller can be a single point of failure or a potential bottleneck. Therefore, in this paper, the traditional SDN controller has been extended and then being embedded in every node, aiming to reduce the risk of losing the connection to the controller and the delay in passing commands from the controller to the devices. This model also enables the benefit of SDN to be utilized in wireless environment. For example, the system resilience has been improved by switching an ongoing connection from a physical interface to another one. The performance of the system was evaluated in a real testbed and the experiment results show that the system which was equipped with the extended SDN controller was a promising solution for handover in heterogeneous wireless network. Specifically, there were a few packet loss when switching an ICMP flow, which has the packet interval of 1ms, between two Wi-Fi interfaces. The packet loss rate was increased up to 37.2% when a handover between Bluetooth and Wi-Fi interface occurred. However, the rate was reduced to below 2.4% when the packet interval was increased from 10ms to 100ms.**

Keywords—extended sdn controller; vertical handover; resilience; openflow rules; heterogeneous wireless network;

I. INTRODUCTION

Ensuring the resilience of networks against natural disasters (i.e., earthquake, fire) or network issues (i.e., link failure, bottleneck) has gained a lot of attention in the 21th century [1, 2, 3]. This means the network operation must be able to endure disruptions within acceptable reduction in service performance. There are a number of approaches to handle this issues including host identity protocol (HIP) [4], media independent handover (MIH) [5], multipath TCP [6] and link aggregation (LA) [7]. In [4, 5], the protocols commonly operate at the Network Layer or higher layers in the OSI model [8], hence, meet a long delay in the handover process, i.e., 1500ms. In [6, 7], the network connection can be maintained in these approaches as it is simultaneously sent over multiple paths or is switched from a faulty link to one or multiple aliveness links. These approaches can be applied to either wired or wireless networks. However, in wireless environments, the considered parameter is not only the network performance, namely, the power consumption of the mobility devices must also be taken into account due to their frequency of using in human-being daily lives. Hence, the mobile device, which has multiple wireless interface such as Wi-Fi, Bluetooth and NFC, commonly uses one interface at any given time. As a result, it is required to utilize standardized interfaces of commodity mobility devices in any attempts to improve the wireless network resilience. Note that almost all of the physical interfaces are commercial products, thus, they may not support LA, MIH or HIP. To this end, Software defined networking technology could be a possible solution.

Software defined networking (SDN) clearly separates the control of networking devices (i.e., router, switch) from themselves, aiming to address the lack of programmability in existing networking architectures [9]. The control of the whole network as well as individual networking device is moved to a centralized network controller which is generally in a detached location with the devices. This model allows the controller to have a global view of the network, hence, enables simpler and easier network management and operation. However, the control path between the devices and the controller could be a single point of failure, as well as a possible bottleneck. Besides, in large networks, there are some delays for the controller‘s commands to reach its devices, thus, it is hard to have a consistent update for the entire network. Moreover, there is very little work focusing on these issues in wireless environments.

There are several related works which apply SDN to the wireless environment in order to avoid link failure. For example, Kanak Agarwal [10] labels all of the potential paths between two hosts and allows an SDN application to select and activate one route. The MAC addresses at two ends are kept unchanged, aiming to enable the traffic to go on another path in case of a link failure. However, this approach does not cover the single point of failure issue of the control path between the controller and its switches. For this purpose, Kien Nguyen [11] used multipath communication to strengthen the controller-switch connections. This approach allows the switch to continue receiving the order from the controller on another path when the default one is failed. Nevertheless, to increase the effectiveness of this method, the placement of the controller should be considered as discussed by Brandon Heller [12]. The resilience of the control path is also improved by putting an

SDN controller on every node [13]. In this approach, an original SDN controller is used locally to handover the traffic between wireless interfaces on one node. After all, this method still requires a global controller to manage the whole network and the author pays attention to the intermediate layer above the virtual point-to-point connection of VPN rather than whole the communication stack.

Different from other studies, in this work, the intermediate layer below the routing (IP) layer is utilized to manage all physical network interfaces. In addition, an extended SDN controller is developed and deployed to every mobility node to cut down any delays in passing commands from the controller to the switch and to make the controller-switch connection safe. The extended controller not only talks with the switch but also has ability to collect the network state and afterward, it makes decision based on the collected information. For instance, the controller decides to switch the traffic on one interface to the other in order to avoid interrupting any running applications if the ongoing connection is getting worse.

The rest of the paper is organized as follows: In Section 2, the traditional SDN controller will be described and be compared with the extended controller. In Section 3, the proposed system will be evaluated. Finally, the paper will be concluded in Section 4.

II. SDN CONTROLLER

A. *Traditional SDN controller*

The SDN architecture consists of three layers as shown in Fig. 1. The lowest layer is the Data plane which comprises forwarding network devices, i.e., switches, in response to forwarding data and collecting statistics. The switches are commonly made by different company and are equipped with their private operation system. Therefore, network administrators must follow the instruction of the company to configure the switch to simply switch traffic from input ports to output ports. In contrast, in SDN networks, switches perform forwarding packets based on flow rules which are kept in a database, or a flow table. A flow rule or a flow entry is inserted, modified or removed by the control plane via a secure communication channel called soundbound interface (SBI). The interface generally uses a protocol named OpenFlow for communication between Data plane and Control plane. The OpenFlow-enabled switches typically ask the controller with a PACKET_IN message when they do not find the matching flow rule for the incoming packets. The controller then answers by asking the switch to assign the actions to the packets. For example, the controller replies with a FLOWMOD message, which is one of the main messages, to modify the state of the switch.

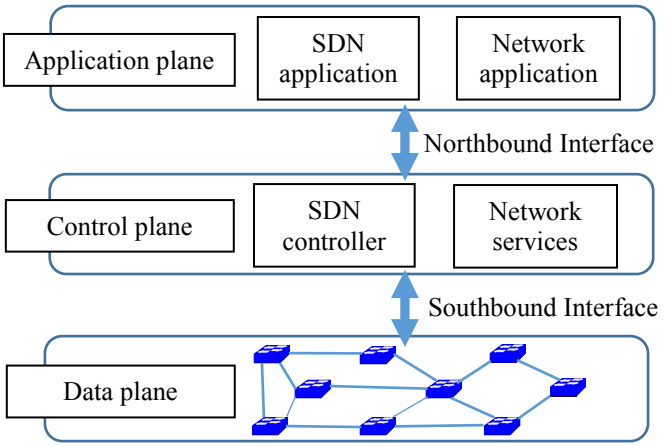

Fig. 1: SDN Architecture

On the Control plane, the controller not only is responsible for controlling network operation, but also listens to the requirement from applications in the Application plane. The controller interprets applications' requirement and passes the corresponded orders to the Data plane. It also provides Network services such as management policy or quality of service to the applications. The controller communicates with the applications via an application-controller interface or northbound interface (NBI). The SDN specification does not standardize NBI. It lets the control software provide their own methods to talk with applications.

B. *Enriched functional SDN controller*

1) *The extended controller design*

Typically, a centralized SDN controller just needs to communicate with OpenFlow switches and handles network events asked by them as it has a global view of the network. However, when the controller and an OpenFlow switch are deployed on the same node, the controller knows nothing about the topology of the entire network. Instead, it has local network information (i.e., an ARP request or RSSI) which cannot be handled by the switch. Thereby, in this work, an extended SDN controller has been developed to allow the controller to handle the switch requests as well as local network behavior.

The control logic of a SDN-based mobile node, in which an original SDN controller and an extended one are embedded, is illustrated in Fig. 2. The node can be a tablet, mobile phone or a laptop. On the top, the original controller and the extended ones are controlling local network behavior for the node. The original SDN controller talks with an OpenFlow-enabled switch, i.e., Open vSwitch (OVS) [14], with OpenFlow protocol. On the right, the extended controller directly collects network information from the physical network interfaces. It is

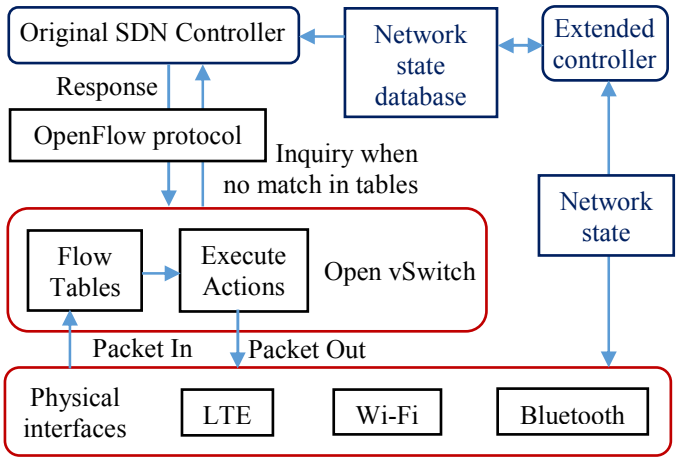

Fig. 2: Control logic of a SDN-based mobile node

developed to be able to customize the network configuration of any interface. The collected information is stored in a database so that the original controller can read and enrich the knowledge about the network topology. In the middle, the OVS navigates the network traffic in and out of the node. It provides one or more virtual bridges which act as one or more middle interfaces sitting between physical interfaces and upper applications. At the bottom, all physical network interfaces are configured to be OVS's switch ports. One bridge can consist of one or more physical interfaces but the interface cannot belong to more than one bridge.

2) Design requirements

a) It must work with any existing SDN controllers

Although existing SDN controllers have their own design with different APIs, the extended part must be able to work with them in SDN networks. To this end, the extended part collects network state and stores them in a database. For flexibility in the deployment, an xml format is picked as a storage. After that, the local SDN controllers, which have no communication channel except the one with switch, can read the network information at any time. If the collected information is big enough, the controller might have a sufficient knowledge of the network topology. It is possible because the extended controller can exchange and update this database to any nodes to which it connects. Table 1 shows the main content of the file, in which, all network configurations are described. For example, the tag Connection is to present the node name and the connection status from the local one to the other nodes. It has two attributes named Name and Status which are marked with X in the Table 1. The second tag, Media, shows the method which the local node is using to connect to the outside. It has three attributes Name, ID and Key. The attribute Name shows the name of the connection, i.e., Wi-Fi or Bluetooth, while ID shows the SSID of the AP or the MAC address of the Bluetooth master and Key contains security string for authentication. On the third row, the tag Bridge describes information of the bridge which is a virtual interface and it has the same number of attribute with the Interface tag which presents physical interfaces. The MAC and IP address for virtual and physical interfaces are recorded, however, these values are defaults ones, they can be changed when the node talks with another node. The invited node normally needs to change its IP address to join the talk. The status of the interface can also be changed to ON or OFF depending on the interface which the node actives and establishes a connection.

TABLE I. NETWORK STATE DATABASE

Tag name	Attributes						
	Name	Switch port	IP	MAC	Status	ID	Key
Connection	X				X		
Media	X					X	X
Bridge	X	X	X	X	X		
Interface	X	X	X	X	X		

b) It must be able to handle the network state

The extended controller must be able to collect network state and analyze them. For example, it collects RSSI values and estimate the distance between nodes. In addition, it needs to have permission to change the state of the physical interfaces. For instance, it can assign new IP address to any interface, or to provide fundamental parameters to a wireless interface so that it will associate with an access point (AP) without user interaction. Thereby, it brings a simpler and easier way to maintain an ongoing session between two hosts when the current physical connection is going to be lost. In theory, to maintain an ongoing session, the same local end-point IP and MAC addresses must be kept through the session. The applications here only see the middle interface created by the OVS. As a result, if the IP and MAC addresses of the middle interface are not changed, the change in the physical interface does not affect the upper applications. Note that wireless base stations generally only allow packets with the source MAC address of NIC that has completed the initial handshake [12]. Consequently, on establishing a new connection, the destination sends out an ARP packet asking for MAC address of the new IP address. The extended controller will automatically send the replies to these requests, however, only selective ARP requests are carefully answered to avoid security risks.

c) It must be able to communicate among nodes

Since the SDN controllers here are local ones, they must talk with the others to exchange knowledge about the network topology. The way they communicate is similar to the way the centralized controllers from different domains talk with each other [10, 11 and 12]. However, the controllers here use a simpler way and exchange different content. For example, when two nodes are connected, one node acts as an inviter and the other node is a peer. The inviter sends a request to exchange message to the peer via a socket connection asking for update their network configuration. After receiving the agreement, the inviter starts to send its data to the peer and wait for the data of the peer to update its local file. The well-described content in the file allows two nodes to keep running applications alive while switching the connection between physical interfaces.

III. EVALUATION

In this section, the evaluation of two handover scenarios are presented: 1) Handover between Wi-Fi and Wi-Fi interfaces, 2) Handover between Wi-Fi and Bluetooth interfaces. Experiments were conducted in a real testbed in which several Linux computers were connected via Wi-Fi or Bluetooth. Each computer was equipped with Core 2 Duo @2.26 GHz and 2GB RAM.

A. Handover between Wi-Fi and Wi-Fi interfaces

The goal of this experiment is to show the feasibility of the extended SDN controller in handover in wireless networks and to provide sample flow entry configurations under different handover scenarios.

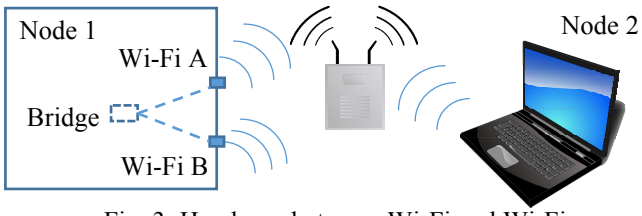

Fig. 3: Handover between Wi-Fi and Wi-Fi interfaces Testbed

In this experiment, node 1, which is an SDN-based mobile node, is talking with a conventional mobile node via an AP as depicted in Fig. 3. The SDN-based one is equipped with two wireless interfaces in which one interface is active and the other is for back-up purpose. Note that when a node has more than two Wi-Fi interface, any packets sent from the node are easily be looped among the interfaces, hence, they must be controlled by the bridge and the default network-manager service must be disabled. Switch port numbers port_wifi_A and port_wifi_B are assigned to the two interfaces, respectively while the switch port number of the bridge is LOCAL. The MAC address of the bridge is borrowed from one interface. It is assumed that the bridge and the Wi-Fi B interface have the same MAC address. On the other hand, the extended controller collects these information and fills into the local database. It also gathers the information about the partner to which the SDN-based node connects. Based on the collected information, OpenFlow rules are described as shown in the Table 2. When the active interface is Wi-Fi B, flow rules are simple as the switch simply forwards the packet to corresponding ports. However, when the active interface is Wi-Fi A, the MAC address of the bridge and the active interface are different, thus, the IP and MAC address in the packet header must be rewritten. The mod-dl-dst and mod-nw-dst fields in the flow actions are used to rewrite the destination MAC and IP address in the packet header. For example, when node 1 sends a packet to node 2, the packet will go through the bridge and be encapsulated with the IP and MAC addresses of the bridge. To detect ICMP packets sent from node 1, the OVS checks incoming packets to see if the packet type is ICMP, the input port in_put is "local" and the source MAC address dl-src is equal to the bridge MAC address bridge-mac. The switch then changes the source and destination MAC and IP addresses to the values of two physical interfaces as if the packets were exchanged between them. This is because a wireless client must associate with the AP before exchanging data. The association time sometimes lasts several seconds, thus, the experiment must be long enough to let the handover event occur during transferring data.

In the Testbed, node 1 sent 4000 ICMP packets to node 2 at the rate of 1ms and handovers took place randomly during the experiment. When a handover took place, the extended controller activated the back-up interface and installed new flow entries into the flow table. After the current interface was shutdown, the flow of ICMP began running over the new interface. The number of packet loss was also observed on the inverted direction. Each experiment was repeated 30 times and the number of packet loss was shown in Figure 5. The figure shows that the highest packet loss rate was 0.3% when switching the ICMP flow from the interface B to the interface A. In the inverted way, there was less packet loss and the number of packet loss was more stable. This is because the interface B has the same MAC address with the bridge, hence, flow rules were simple and the system spent less time to switch the flow.

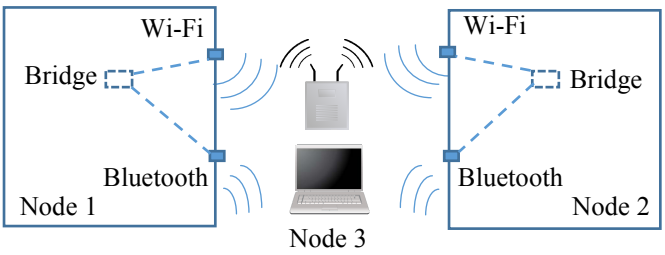

Fig. 4: Handover between Wi-Fi and Bluetooth interfaces Testbed

TABLE II. FLOW RULES

Direction	*Flow rules*	*Flow actions*
From interface A to B	in_port=local in_port= port_wifi_B	Output: port_wifi_B Output: local
From interface B to A	in_port = local, dl-src = bridge-mac, type = icmp in_port = port_wifi_A, dl-dst = wifi-A-mac, type = icmp	mod-dl-dst=wifi-dst-mac, mod-nw-dst=wifi-dst-IP, mod-nw-src=wifi-A-IP, mod-dl-src=wifi-A-mac, output: port_wifi_A mod-dl-dst=bridge-mac, mod-nw-dst= bridge -IP, output: LOCAL

B. Handover between Wi-Fi and Bluetooth interfaces

The goal of this section is to evaluate the performance of the system in the vertical handover between Wi-Fi and Bluetooth interfaces. Since two physical interfaces use different ways to connect, switching traffic between them requires coordination between nodes. The following experiment presents how to synchronize the switching process in two nodes as well as flow rule description.

Figure 4 shows the scenario for the vertical handover experiment. In the figure, two SDN-based mobile nodes equipped with one Wi-Fi and one Bluetooth interface are communicating. It is assumed that the nodes can be either members in the same Bluetooth piconet formed by another conventional mobile node or two wireless clients which connect to the same AP. All physical interfaces are controlled by the OVS and switch port numbers are assigned to them. The OVS chooses the MAC address for the bridge, assumed that it takes the value from the Bluetooth interface. These information is collected and recorded in the local database which has the same structure as the one in previous experiment. The system also uses the same way to define flow rules to rewrite IP and MAC address in the packet header. However, this scenarios requires two local controller to talk to the other when it is ready to switch traffic from the current interface to the other one. Note that the Wi-Fi interface has a higher speed than the Bluetooth one, hence, it is difficult to separate the number of

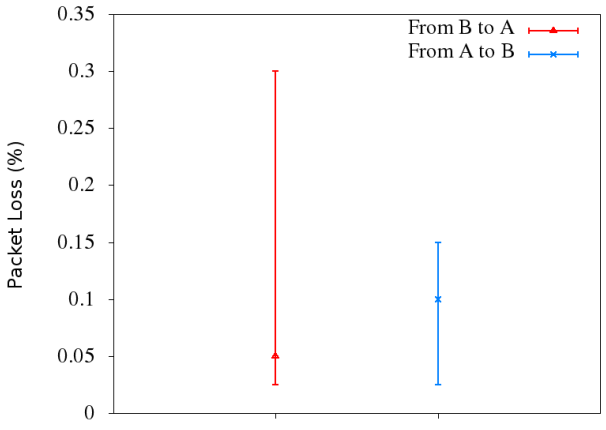

Fig.5: Packet Loss Counts in Handover between Wi-Fi and Wi-Fi interfaces

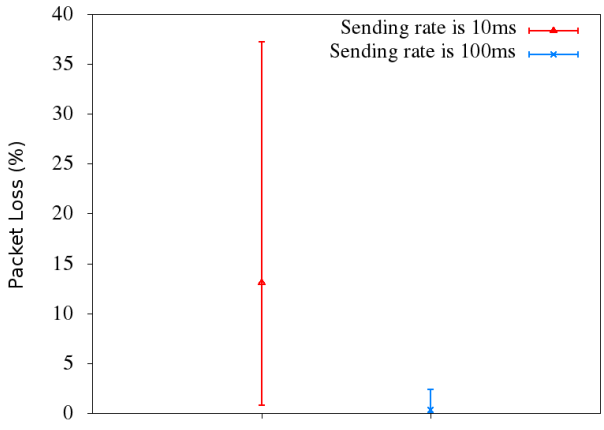

Fig.6: Packet Loss Counts in Handover between Wi-Fi and Bluetooth interfaces

packet loss because of reducing the connection bandwidth and the packet loss during handover. Consequently, in this experiment, the handover only happens in the direction from Bluetooth interface to Wi-Fi interface. When the Bluetooth connection is going to be lost, i.e., two nodes are out of the Bluetooth communication range, the ongoing traffic is automatically shifted to the Wi-Fi connection.

In the first experiment, 1000 ICMP packets were exchanged between two nodes at the rate of 10ms and handovers took place randomly during the experiment. The experiment was repeated 30 times with random handover and the number of packet loss are presented in Fig. 6. Figure 6 shows that the packet loss is up to 37.2%, or the maximum number of packet loss is 372 packets over 1000 sent ones. This rate is reduced when packets are sent with a slower speed. The highest rate of packet loss now is only 2.4% when the packet interval is 100ms. The results confirmed that the extended controller enhanced the system resilience since it can automatically switch the traffic.

IV. CONCLUSION AND FUTURE WORK

The goal of this work is to increase the resilience of the system in wireless networks and an extended SDN controller has been proposed. In the system, the controllers were deployed in a distributed manner in order to reduce the risk of losing the control path between the SDN controllers and OpenFlow switches. The controller then collected network state on the node and gained the knowledge about the network topology by discussing with any node to which it connected. The collected network information was stored in a database, i.e., an xml file so that the information was sharable with any traditional SDN controllers. The performance of the system was evaluated under different scenarios. The evaluation results confirmed that the extended SDN controller can maintain an ICMP flows in heterogeneous wireless networks. However, this work did not cover the security of the system and as it is in the primary state, the system did not handle all existing protocols.

The future work will concentrate on optimizing the handover algorithm to reduce packet loss rate in the vertical handover and consider the handover in multi-hop networks.

REFERENCES

[1] K. Fukuda, M. Aoki, S. Abe, Y. Ji, M. Koibuchi, M. Nakamura, S. Yamada, and S. Urushidani, “Impact of Tohoku earthquake on R&E network in Japan,” in Proceedings of the Special Workshop on Internet and Disasters, 2011.

[2] Quang Tran Minh, Kien Nguyen, and Shigeki Yamada, "DRANs: Resilient Disaster Recovery Access Networks", Proc. IEEE IWFIT 2013, in conjunction with IEEE COMPSAC 2013, Kyoto, Japan, July 2013.

[3] Shoichi Senda, Kien Nguyen, and Shigeki Yamada, "Requirements for Resilient Information and Communication Technology", Proc. International Workshop on Engineering Complex Distributed Systems (ECDS-2013),Taichung, Taiwan, July 2013.

[4] S. Namal, J. Pellikka, and A. Gurtov, “Secure and multihomed vehicular emtocells,” in 75th Vehicular Technology Conference (VTC Spring). IEEE, 2012, pp. 1–5.

[5] G. Lampropoulos, A. Salkintizis, and N. Passas, “Media Independent Handover for Seamless Service Provision in Heterogeneous Networks,” IEEE Commun. Mag., Jan. 2008, pp. 64–71.

[6] Costin Raiciu, Christoph Paasch, Sbastien Barr, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley, “How Hard Can It Be? Designing and Implementing a Deployable MUltipath TCP”. In Proc. USENIX NSDI, 2012.

[7] IEEE Std 802.1AX-2008.
http://standards.ieee.org/findstds/standard/802. 1AX-2008.html, 2008.

[8] H. Zimmermann, OS1 Reference Model - The IS0 Model of Architecture for Open Systems Interconnection, IEEE Transactions on Communications COM-28, No. 4: April 1980.

[9] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner, “OpenFlow: enabling innovation in campus networks”. SIGCOMM Comput. Commun. Rev., 38(2), March 2008.

[10] Kanak Agarwal, Colin Dixon, Eric Rozner, and John Carter, "Shadow MACs: scalable label-switching for commodity ethernet" In Proceedings of the third workshop on Hot topics in software defined networking (HotSDN '14). ACM, New York, USA, 2014.

[11] Kien Nguyen, Quang Tran Minh, and Shigeki Yamada, "Increasing Resilience of OpenFlow WANs using Multipath Communication" Proc. IEEE ICITCS2013, Macau, China, December 2013.

[12] Brandon Heller, Rob Sherwood, and Nick McKeown, “The controller placement problem”. In Proc. 1st Workshop on Hot topics in software defined networks, HotSDN ’12, pages 7–12, 2012.

[13] Ryan Izard, Adam Hodges, Jianwei Liu, Jim Martin, Kuang-Ching Wang, Ke Xu, “An openflow testbed for the evaluation of vertical handover decision algorithms in heterogeneous wireless networks”. In the 9th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, TridentCom 2014, Guangzhou.

[14] Open vSwitch. http://openvswitch.org/.