Experiments with OpenFlow and IEEE802.11 Point-to-Point Links in a WMN

Michael Rademacher*, Florian Siebertz†, Moritz Schlebusch‡ and Karl Jonas§
Department of Computer Science, Bonn-Rhein-Sieg University of Applied Sciences
Grantham-Allee 20, 53757 Sankt Augustin
Email: *michael.rademacher@h-brs.de, †florian.siebertz@inf.h-brs.de, ‡moritz.schlebusch@inf.h-brs.de, §karl.jonas@h-brs.de

***Abstract*—Software Defined Networking (SDN) and Wireless Mesh Networks (WMNs) evolved to be sophisticated technologies used in a variety of applications. However, a combined approach called Wireless Mesh Software Defined Network (wmSDN) has not been widely addressed in the research community. Our idea in this field consists of WiFi-based point-to-point links managed by the OpenFlow protocol. We investigate two different issues regarding this idea. First, which WiFi operational mode is suitable in an OpenFlow managed broadcast domain? Second, does the performance decrease compared with other routing or switching principles? Therefore, we set up a real-world testbed and a suitable simulation environment. Unlike previous work, we show that it is possible to use WiFi links without conducting Media Access Control (MAC) address rewriting at each hop by utilizing the 4-address-mode.**

***Keywords*–SDN; OpenFlow; WDS; wmSDN; SDWN; WiFi**

I. INTRODUCTION

SDN and OpenFlow started as an academic experiment but emerged to a paradigm that challenges the limitations of current static network infrastructures [1]. SDN related research and applications mainly focus on wired infrastructures. However, recent work discusses the advantages also in the context of wireless networks, especially in the backhaul segments [2]. WMNs have been a research topic for several years and they are used successfully for last-mile connectivity [2]. In this case, a WMN is used with multiple radios and Commercial Off-the-Shelf (COTS) WiFi transmitters over long-distance to form what is called a WiFi-based Long Distance (WiLD) network [3].

Our motivation is a combination of SDN and WiLD as already introduced in Hadzic et al. [2]. The following are - by no means exhaustive - examples for benefits of such a combination:

- If SDN based infrastructure is already utilized in the core part of a network it feels natural to extend it further into the last-mile.
- SDN maintains a centralized network state. This provides the flexibility to configure, manage, secure and optimize the network resources using SDN applications.
- In WMNs, traffic pattern can change on a relatively small time-scale. With a centralized SDN based structure, the network can react to these changes based on global knowledge.
- With an increased number of mesh nodes in the network, multiple gateways providing interconnection to public networks become mandatory. SDN applications can handle these multiple gateways and configure the flows accordingly.

In this work, we share first experiences running commodity IEEE 802.11 hard- and software in combination with a recent OpenFlow implementation. We found that a major challenge for the usage of, e.g., the Open vSwitch (OVS) in combination with WiFi (IEEE 802.11) is the handling of MAC addresses. In fact, the usage of wireless links is currently marked as not supported by the OVS. The documentation summarizes this issue as follows: “Wireless base stations generally only allow packets with the source MAC address of NICs that completed the initial handshake. Therefore, without MAC rewriting, only a single device can communicate over a single wireless link.” [4].

The remainder of this paper is structured as follows. In Section II, we summarize related work in the context of wmSDN. In Section III, the principles of combining OpenFlow and OVS, as well as the different possible addressing modes in IEEE 802.11 are described. Section IV is twofold, presenting our methodology: First, we describe a small testbed based on COTS hardware. Afterwards, we introduce a wmSDN simulation environment running the same OpenFlow software components used in the testbed. In Section V, we describe our experiments and the results. This paper closes with a summary and an outlook on future work items.

II. SOFTWARE DEFINED WIRELESS NETWORKS

SDN is an architecture where network control is decoupled from the task of forwarding packets. This is achieved by removing the decision making process regarding the handling of packets from every single device. A centralized controller entity is deployed instead, conducting decisions for all devices in the network [1].

Software Defined Wireless Networks (SDWNs) or wmSDNs are beginning to form themselves on top of the experiences gathered in wired networks. What differs is the need for handling the new layer of complexity that is added through wireless interfaces. The terms wmSDN or SDWN are used in two different ways in the research community. SDWN describes a SDN controlled network in which the backbone is typically wired but interconnects several wireless access networks. The term wmSDN is often used to describe a wireless SDN backbone network. In the following, we summarize important contributions in both research fields.

Different generic architectures with the goal of implementing SDN into existing wireless technologies and networks have been proposed [5][6][7]. In Hadzic et al. [2], a concept for integrating SDN into a wireless backhaul infrastructure is presented. A more detailed thought is given to how control- and data-plane must be adapted in order to provide the setup and configuration of diverse wireless interfaces. The authors propose additional modules in the control plane that gather information regarding the spectrum management and link capacity in the entire network via an extended southbound interface.

In Dely et al. [8], the authors present an IEEE 802.11 architecture where packets are forwarded between virtual Access Points (APs) and stations based on MAC layer processing via OpenFlow. This enables applications like dynamic spectrum use, on-demand scaling, and improved roaming. Only the access part of the network utilizes wireless SDN technology in this case.

One possibility of using OpenFlow alongside link configuration mechanisms is the extension of the existing OpenFlow protocol with new functionality. Guimaraes et al. [9] propose additional control messages to be included into the protocol for querying the link state and setting interface parameters. The controller has to be extended in order to handle the additional configuration aspects. Nascimento et al. [10] take a similar approach by deploying a wmSDN that extends the OpenFlow protocol capabilities with new control messages. Additionally, it is proposed to include the IEEE 802.11 MAC header into the list of headers that can be examined by OpenFlow to handle forwarding on each hop. The authors identify that it is necessary to manipulate the source and destination addresses of each packet as it traverses through the network. The proposed solution is to exchange the addresses at each hop. This is taken into account by adding new actions to the flow table.

Another approach is presented in Dely et al. [11]. The OpenFlow protocol and controller remain unchanged, while the wireless configuration and monitoring is done via a separate monitoring and control server that communicates with agents on each device. This server handles the topology, associations between stations, and monitors changes in order to adapt the network accordingly. All control data, including the OpenFlow protocol, is managed through a separate Service Set Identifier (SSID) and the Optimized Link State Routing (OLSR) protocol. The packet forwarding problem mentioned above is addressed in the same way through manipulating the MAC addresses on each hop.

Hurtado-Borràs et al. [12] examine a wireless backhaul with Multi-Radio (MR) wireless switches. The controller is attached in-band, with a preconfigured path through the use of one parent interface on each node, indicating the next hop towards the controller. The issue of wireless status and configuration is addressed by extending the interface between controller and node accordingly.

The OVS can be used to bring OpenFlow capabilities onto a large number of devices. Although it is not meant for wireless networks explicitly, a number of projects have succeeded in adapting it for this scenario. Lee et al. [13] propose an open-source platform for wireless research on the basis of OVS. Mihailescu et al. [14] utilize the OVS in their wireless nodes combined with a local controller that handles the local status and configuration. Detti et al. [15] use the OVS to build OpenFlow capable, wireless devices. The controller is connected in-band via the same SSID and in combination with OLSR to route OpenFlow control messages. Similar to the other works, MAC-rewriting is used in order to enable packet forwarding. In Lima et al. [16], the authors deploy the OVS on OpenWRT driven commodity hardware similar to our approach, comparing operation with and without OpenFlow and between different controllers.

Table 1. Address field combinations of IEEE 802.11.

Control Bits		Address Field				
To DS	From DS	1	2	3	4	Usage
0	0	RA=DA	TA=SA	BSSID	N/A	ad-hoc
0	1	RA=DA	TA=BSSID	SA	N/A	AP
1	0	RA=BSSID	TA=SA	DA	N/A	Station
1	1	RA	TA	DA	SA	WDS

III. OpenFlow Based Switching and IEEE 802.11

One of the main concepts of SDN is that the network appears to the applications and policy engines located at the centralized controller as a single, logical switch [17]. Since IEEE 802.11 implies a different addressing principle, a seamless integration is proving challenging, as described in Section II. MAC rewriting for every packet is proposed by other research to tackle this issue and enabling packet forwarding via IEEE 802.11 links on OpenFlow capable switches [15][11]. In this section, we investigate the basics for an operation of different IEEE 802.11 modes on OpenFlow enabled switches.

When running a layer 2 OpenFlow switching application, an OpenFlow network can operate similar to a common switched network. The manipulation of flows is based on the ingress port and the destination MAC address. However, in SDN, the initial decision for a manipulation is made at the centralized controller and not on the switches themselves.

The following steps summarize the process of a packet arriving at an OpenFlow enabled switch, i.e., the OVS. If no existing flow matches the packet, it is sent to the controller. Different actions are applied based on the source and destination address. If the source MAC address is unknown to the controller, the controller learns it by binding the address to the ingress port on the switch. If the destination MAC address is unknown to the controller, the controller initiates a flood on all remaining ports of the switch. If the destination MAC address is known to the controller after learning it from the source MAC address of a different packet, the controller installs a flow rule on the switch. When a flow has been installed on the switch successfully, future packets are processed on the switch itself. More details about this example and additional applications can be found in the work [18].

The MAC addressing in IEEE 802.11 differs from IEEE 802.3. The current IEEE 802.11 standard [19] distinguishes between five different address types:

- Transmitter Address (TA): The individual MAC address that identifies the station that has transmitted onto the wireless medium.
- Receiver Address (RA): The individual or group MAC address that identifies the intended immediate recipient on the wireless medium.
- Source Address (SA): The individual MAC address from which the transfer was originally initiated.
- Destination Address (DA): The individual or group MAC address that identifies the final recipients.

- Basic Service Set Identifier (BSSID): The address of a set of stations that are associated successfully.

Up to four different address fields can be used at the same time and four different combinations are specified in the current standard as listed in Tab. 1. The number of address fields used and their contents are controlled by two bits in the MAC header - To DS and From DS. In the last case, To DS and From DS are set to "1", resulting in all four address fields being used. The first two address fields are set to RA and TA respectively, while the last two are set to the DA and SA. This mode of operation is commonly known as 4-address-mode or Wireless Distribution System (WDS).

IV. Methodology

We utilize two different evaluation environments for our use-case of a wmSDN: A real world testbed using COTS hardware and a simulation environment. Despite the different nature and scope of both approaches, we ensure that they share the same configuration and software as far as possible. All wireless links use the IEEE 802.11n standard, either in the 2.4 GHz or 5 GHz band. The channel width is limited to 20 MHz and the Modulation and Coding Scheme (MCS) is fixed to index number 7. This MCS results in no Multiple-Input and Multiple-Output (MIMO) usage and a maximum physical-layer bitrate of 65 Mbps. The main reasons for the fixed MCS is to avoid undesired effects of a rate-controller. At the moment of writing, MIMO is not supported in the used simulation software ns-3 3.24. Therefore, we also limit our experiments to Single-Input and Single-Output (SISO) in the real-world testbed.

Two main software components are necessary for our network evaluation: An OpenFlow capable switch and a controller. Ryu [20] in version 3.26 is used as the OpenFlow controller, running the preinstalled OpenFlow 1.3 layer two switching module operating on a i686 fanless mini-pc system running Ubuntu 14.04. For the switching part, we choose the latest version 2.3.90 of the OVS [4].

Both components are used on the actual devices and in the simulation environment. All devices are connected out-of-band to the controller via a dedicated wired control network. Individual configurations for the testbed and the simulation are summarized in the following subsections. Additional documentation, configurations and materials are available to the research community on our website [21].

A. Testbed

The testbed consists of TP-Link WDR4300 routers version 1.7, running OpenWRT 15.05 "Chaos Calmer" (rev. 47499, Kernel 3.18.23). The OVS enables OpenFlow on the routers which we therefore call wireless switches. The wireless configuration is conducted in the respective OpenWRT configuration files. To enable the 4-address-mode, we set the option "wds" on the wireless interfaces as enabled.

B. Simulation

For our simulation, we use ns-3 in combination with the Mininet framework [22]. The main reason for this choice is ns-3's limited ability to enable OpenFlow and our desire to use the same OpenFlow components in the testbed and in the simulation. We build upon the work done in [23]. The author implements Mininet and ns-3 enhancements, enabling the ns-3 modeling of links between container-virtualized Mininet nodes. The connection is done by combining Linux virtual network devices, called TAP-devices, and a ns-3 module called TAP-bridge. In our case, the TAP-device connects to the ns-3 process by using a file descriptor. When packets generated at a Mininet host arrive at the ns-3 process, they are forwarded by the TAP-bridge to the ns-3 net device and transmitted across the emulated ns-3 WiFi channel using the Friis propagation loss model. Two additional modifications are necessary to build the desired simulation environment: First, the adaption of a ns-3 patch, enabling wireless links in 4-address-mode [23]. Second, preparation of wmSDN devices based on Mininet nodes.

V. Experiments

In the first part of this section, we evaluate the forwarding issue for IEEE 802.11 links in an OpenFlow managed broadcast domain. Afterwards, we evaluate the performance of a wmSDN in a simple multi-hop chain topology.

A. IEEE 802.11 and OpenFlow

In this section, we present the results obtained when integrating WiFi links operating in different modes, as shown in Tab. 1, into a broadcast domain managed with OpenFlow. For simplicity, we reduce this experiment to a minimal topology. We use two of our OpenFlow enabled wireless switches (S1,S2) described in Subsection IV-A and two hosts (H1,H2) connected to the OpenFlow managed Ethernet ports. To evaluate the packet exchange, we use a dedicated sniffing system running a WiFi card in monitor mode.

The packet exchange for the experiments is shown in Fig. 1. Exemplary, we choose the Address Resolution Protocol (ARP) to display the behavior. ARP-request are sent between two hosts (H1,H2) over one IEEE 802.11 hop (S1,S2). Three different modes for the IEEE 802.11 interfaces are used independently. The labels on the arrows show the MAC addresses:

- Destination (dst) and source (src) for IEEE 802.3
- 1–4 for IEEE 802.11 according to Tab. 1

All interfaces are managed by the OVS and the OpenFlow protocol, while the wireless switches are controlled out-of-band by the OpenFlow controller. In all experiments, H1 asks for the hardware address of H2. The ARP-request arrives at the OVS on S1 and is forwarded to the wireless interface. Depending on the WiFi mode, the previous dst and src addresses are matched differently to the 802.11 header.

The first experiment is conducted using what is commonly known as ad-hoc mode on S1 and S2. In this mode, dst and src addresses are matched to the first two addresses in the 802.11 packet header. The Independent Basic Service Set (IBSS) is added in address field 3. Due to the fact that the dst address is a broadcast address, the receiving wireless station accepts the packet. The OVS running on S2 floods the packet to H2 where the ARP-reply is generated, which is again sent back to S2. The ARP-reply is sent on the wireless interface with the first address field set to H1. This results in the receiving wireless interface on S1 not processing the frame, as it is not the target. The communication is unsuccessful.

The second experiment is conducted using the AP mode on S1 and the station mode on S2. Again, due to the broadcast

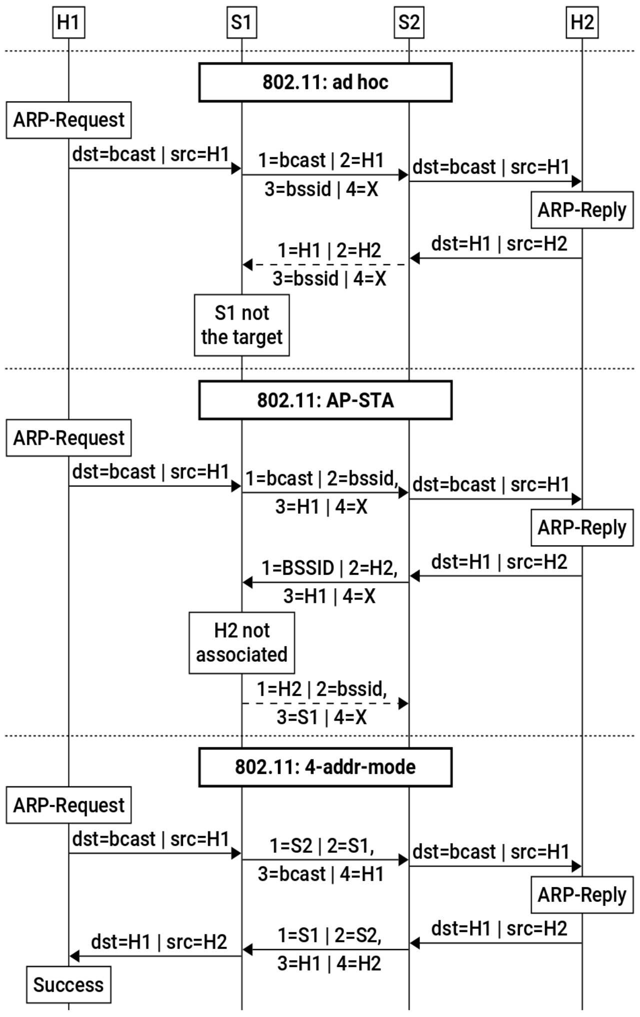

Figure 1. ARP-requests between two hosts are transmitted over one OpenFlow controlled wireless link in different modes.

address, the frame arrives at H2, which generates the ARP-reply. The ARP-reply is forwarded by S2. The transmission is unsuccessful because address field 2 in the 802.11 header is set to H2. The OVS forwards the packet on the wireless port, but the wireless AP does not accept the packet, since H2 is not associated with the AP. In fact, the AP sends a deauthentication frame with the destination set to H2. This deauthentication is ignored by S2.

The last experiment shows the results using the 4-address-mode on S1 and S2. Due to the usage of four addresses, the communication is successful. In fact, the first two addresses in the IEEE 802.11 header can be described as link scope while the last two addresses retain the original communication between the two hosts. This behavior is transparent to the OVS.

B. *Performance Measurements*

In this subsection, we present different performance measurements conducted in our testbed and in the simulation environment. All configuration and parameters from the testbed are adapted to the simulation as far as possible. We compare the following cases:

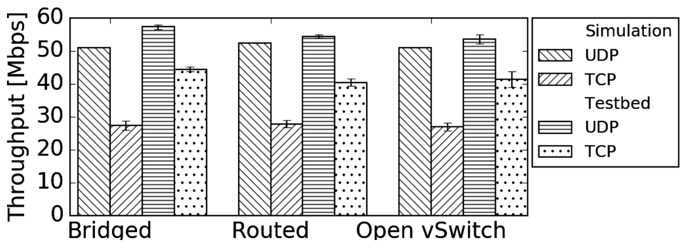

Figure 2. Results of the performance test using four OpenFlow enabled wireless routers and the simulation environment.

- Linux kernel *bridging*, as used by the IEEE 802.11s mesh protocol [24] with WiFi interfaces in 4-address-mode.
- IP based Linux kernel *routing*, as used by different routing protocols in Multi-Radio Multi-Channel (MR-MC) WMNs [25] with WiFi interfaces in AP and station mode.
- An OpenFlow managed network, as shown in the previous section using WiFi interfaces in 4-address-mode.

In this experiment, conducted indoors and under laboratory conditions, four dual-band wireless switches are arranged in the corners of a square plane spanning 3x3 meters in a clear line of sight environment. The four devices are configured to form a chain topology, utilizing two 5 GHz and one 2.4 GHz links on independent channels. One server and one client are connected to the first and the last router via Ethernet. The performance measurements are conducted with artificial User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) traffic using the well-known tool iPerf. Each test is executed five times, lasting 5 minutes in total. The same topology is implemented in the developed simulation environment. For real-world tests, e.g., Rademacher et al. [26] demonstrate the effects that are accompanied by using IEEE 802.11 over longer distances and their influence on parameters like throughput and jitter.

The results of our tests are shown in Fig. 2. Due to the fixed MCS, the maximum physical data rate is bounded by 65 Mbps. First, we describe the results from the testbed and afterwards from the simulation environment.

In our testbed scenario, we measure a maximum unidirectional UDP rate of 57.2 ± 0.7 Mbps for the bridged mode test. This throughput is close to a saturation of all wireless links considering the 802.11n MAC overhead. A slight performance decrease for the routed (54.3 ± 0.4 Mbps) and OVS (53.4 ± 1.3 Mbps) case has been obtained. For the Linux bridged mode, packet headers are not evaluated or changed. In the case of IP based routing, an IP table lookup and address rewrite is necessary. In the case of OVS, for every packet, the flow tables are evaluated.

Similar observations can be made for TCP, although the throughput is consistently lower due to the incurring overhead (bridged: 44.3 ± 0.7 Mbps, routed: 40.4 ± 1.1 Mbps, OVS: 41.3 ± 2.3 Mbps). Performance differences between bridging, routing and OVS have also been reported by other research in the case of wired networks [27]. However, the measurements show that the performance of OVS managed wireless links using the 4-address-mode is comparable to already established approaches.

The measured simulated throughput differs in some aspects from the results obtained in the testbed environment. Noticably, all three cases show the same results for UDP ($\approx 51Mbps$) and TCP ($\approx 27Mbps$). Compared to the testbed, UDP throughput is slightly, and TCP traffic is considerably lower. This behavior could indicate a performance bottleneck within the simulation environment. Due to it's complex architecture, further research is needed in order to identify the source of the bottleneck.

VI. Summary

In this work, we present the difficulty of using wireless links and OpenFlow enabled switches. Because of the way packet forwarding is realized in OpenFlow, just adding the respective interface to the OVS is not sufficient. To evaluate the solution of a wmSDN multi-channel network, we developed a testbed and simulation environment. The same OpenFlow software components were used in both environments.

The results show that with ad-hoc and AP Station mode and the fact that packet headers are not modified at each hop, using OpenFlow does not work with the 802.11 encapsulation. We successfully show that using the 4-address-mode on the wireless links resolves this issue. Thus, it is possible to preserve source and destination addresses from the IEEE 802.3 frames and to insert the respective WiFi address information per link automatically. Unlike previous work, we demonstrate the general applicability of using OpenFlow in an IEEE 802.11 based wmSDN without the need for manually overwriting MAC addresses at each hop.

To show the feasibility of this solution, we compare the throughput of OpenFlow controlled wireless bridging with 4-address-mode enabled to traditional bridging and classic routing in a simulation environment and a testbed. In the testbed enviroment, using COTS hardware, the performance difference between all three solutions are negligible. In our opinion this technology combination provides interesting capabilities in different use-cases - for example in a WiLD network used for internet provisioning in rural areas.

A. Future Work

Several important issues remain unaddressed and need further investigation. Especially for a complete wmSDN solution suitable to function as a WiLD. While we have limited the presented research in this work to an evaluation in a laboratory testbed, the suitability for WiLD networks still needs to be evaluated. This includes in-band connections of the controller, a discovery algorithm for the wireless SDN switches, monitoring of wireless link states, and multiple internet gateways. Instead of using the simple OpenFlow switching application, a more sophisticated routing approach based on this data is desired.

Acknowledgment

This work has been funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (Foerderkennzeichen 16KIS0332, Virtualisierung von Netzfunktionen in Funknetzen auf der Basis von Software Defined Networks). The authors alone are responsible for the content of this paper.

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey,” in Proceedings of the IEEE, vol. 103, no. 1, Jan. 2015, pp. 14–76.

[2] S. Hadzic, C. Niephaus, O. G. Aliu, G. Ghinea, and M. Kretschmer, “Wireless Back-haul: a software defined network enabled wireless Back-haul network architecture for future 5G networks,” IET Networks, vol. 4, no. 6, Nov. 2015, pp. 287–295.

[3] M. Rademacher, “Performance estimation and optimization of the IEEE802.11 MAC layer for long distance point-to-point links,” Hochschule Bonn-Rhein-Sieg, Tech. Rep., 2015, [Retrieved: Sep. 2016]. [Online]. Available: http://opus.bib.hochschule-bonn-rhein-sieg.de/opus-3.3/volltexte/2015/30

[4] The Linux Foundation, “Open vSwitch – Frequently Asked Questions,” [Retrieved: Sep. 2016]. [Online]. Available: https://github.com/openvswitch/ovs/blob/master/FAQ.md

[5] C. Bernardos, A. De La Oliva, P. Serrano, A. Banchs, L. Contreras, H. Jin, and J. Zuniga, “An Architecture for Software Defined Wireless Networking ,” Wireless Communications, IEEE, vol. 21, no. 3, June 2014, pp. 52–61.

[6] S. Tomovic, K. Yoshigoe, I. Maljevic, M. Pejanovic-Djurisic, and I. Radusinovic, “SDN-based concept of QoS aware heterogeneous wireless network operation,” in Telecommunications Forum Telfor (TELFOR), 2014 22nd, Nov. 2014, pp. 27–30.

[7] H. Ali-Ahmad, C. Cicconetti, A. De la Oliva, V. Mancuso, M. R. Sama, P. Seite, and S. Shanmugalingam, “An sdn-based network architecture for extremely dense wireless networks,” in Future Networks and Services (SDN4FNS), 2013 IEEE SDN for. IEEE, 2013, pp. 1–7.

[8] P. Dely, J. Vestin, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo, “CloudMAC - An OpenFlow based Architecture for 802.11 MAC Layer Processing in the Cloud,” in Globecom Workshops (GC Wkshps), 2012 IEEE, Dec. 2012, pp. 186–191.

[9] C. Guimaraes, D. Corujo, and R. Aguiar, “Enhancing OpenFlow with Media Independent Management Capabilities,” in Communications (ICC), 2014 IEEE International Conference on, June 2014, pp. 2995–3000.

[10] V. Nascimento, M. Moraes, R. Gomes, B. Pinheiro, A. Abelem, V. Borges, K. Cardoso, and E. Cerqueira, “Filling the Gap Between Software Defined Networking and Wireless Mesh Networks,” in Network and Service Management (CNSM), 2014 10th International Conference on, Nov. 2014, pp. 451–454.

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

[12] A. Hurtado-Borràs, J. Pala-Solé, D. Camps-Mur, and S. Sallent-Ribes, “SDN wireless backhauling for Small Cells,” in 2015 IEEE International Conference on Communications (ICC). IEEE, 2015, pp. 3897–3902.

[13] S. Lee, Y.-H. Kim, and S. Yang, “Open-source Wireless Switch for Experimental Research Vitalization for B4G/5G Networks,” in Ubiquitous and Future Networks (ICUFN), 2014 Sixth International Conf on, July 2014, pp. 125–126.

[14] M. Mihailescu, H. Nguyen, and M. Webb, “Enhancing Wireless Communications with Software Defined Networking,” in Military Communications and Information Systems Conference (MilCIS), 2015, Nov. 2015, pp. 1–6.

[15] A. Detti, C. Pisa, S. Salsano, and N. Blefari-Melazzi, “Wireless Mesh Software Defined Networks (wmSDN),” in Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on, Oct. 2013, pp. 89–95.

[16] L. Lima, D. Azevedo, and S. Fernandes, “Performance evaluation of openflow in commodity wireless routers,” in Network Operations and Management Symposium (LANOMS), 2015 Latin American. IEEE, 2015, pp. 17–22.

[17] O. N. Foundation, “Software-defined networking: The new norm for networks,” ONF White Paper, 2012.

[18] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: enabling innovation in campus networks,” ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, 2008, pp. 69–74.

[19] IEEE Standards Association and others, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007), Feb. 2012, pp. 1–2793.

[20] NN, “Ryu - a component-based software defined networking framework,” [Retrieved: Sep. 2016]. [Online]. Available: https://osrg.github.io/ryu/

[21] ——, “MC-Lab Homepage,” [Retrieved: Sep. 2016]. [Online]. Available: http://mc-lab.de

[22] ——, “Mininet - An Instant Virtual Network on your Laptop,” [Retrieved: Sep. 2016]. [Online]. Available: http://mininet.org/

[23] P. Jurkiewicz, “Link modeling using ns 3,” https://github.com/mininet/mininet/wiki/Link-modeling-using-ns-3, [Retrieved: Sep. 2016].

[24] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, “IEEE 802.11s: The WLAN Mesh Standard,” IEEE Wireless Communications, vol. 17, no. 1, Feb. 2010, pp. 104–111.

[25] A. Raniwala and T.-c. Chiueh, “Architecture and algorithms for an ieee 802.11-based multi-channel wireless mesh network,” in Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., vol. 3. IEEE, 2005, pp. 2223–2234.

[26] M. Rademacher, M. Chauchet, and K. Jonas, “A Token-Based MAC For Long-Distance IEEE802. 11 Point-To-Point Links,” ITG-Fachbericht-Mobilkommunikation–Technologien und Anwendungen, 2016.

[27] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, “OpenFlow Switching: Data Plane Performance,” Communications (ICC), 2010 IEEE International Conference on, 2010, pp. 1–5.