OpenFlow for Wireless Mesh Networks

Peter Dely, Andreas Kassler
Karlstad University, Sweden
peter.dely@kau.se, andreas.kassler@ieee.org

Nico Bayer
Deutsche Telekom Laboratories, Germany
nico.bayer@telekom.de

***Abstract*—Several protocols for routing and forwarding in Wireless Mesh Networks (WMN) have been proposed, such as AODV, OLSR or B.A.T.M.A.N. However, providing support for e.g. flow-based routing where flows of one source take different paths through the network is hard to implement in a unified way using traditional routing protocols. OpenFlow is an emerging technology which makes network elements such as routers or switches programmable via a standardized interface. By using virtualization and flow-based routing, OpenFlow enables a rapid deployment of novel packet forwarding and routing algorithms, focusing on fixed networks. We propose an architecture that integrates OpenFlow with WMNs and provides such flow-based routing and forwarding capabilities. To demonstrate the feasibility of our OpenFlow based approach, we have implemented a simple solution to solve the problem of client mobility in a WMN which handles the fast migration of client addresses (e.g. IP addresses) between Mesh Access Points and the interaction with re-routing without the need for tunneling. Measurements from a real mesh testbed (KAUMesh) demonstrate the feasibility of our approach based on the evaluation of forwarding performance, control traffic and rule activation time.**

***Index Terms*—Architecture, Routing, Handover**

I. Introduction

A Wireless Mesh Network (WMN) is a multi-hop wireless network, in which stationary mesh routers wirelessly relay traffic on behalf of other mesh routers or client stations and thereby form a wireless backbone. For enabling potentially mobile clients (such as notebooks or smartphones) to connect to the mesh, Mesh Access Points (MAP) provide a standard WLAN interface and act as mesh routers which forward traffic towards the destination via other mesh routers or the gateway, which connects the mesh to the Internet.

Routing in mesh networks has been a very active research field. Different approaches to routing have been investigated. Several approaches have an ad-hoc networking flavor (e.g. AODV [15] or B.A.T.M.A.N. [10]), while others are more influenced by classical LAN routing protocols (e.g. OLSR [6]). The functionality of those routing protocols is fairly limited and extensions are difficult. For example, flow-based routing, where flows take different paths through the network to reach one destination is hard to implement with traditional routing protocols. However, such flow-based routing enables interesting use cases such as load-balancing. Another challenging task is to enable client mobility in the mesh network. Here, a routing protocol needs to handle the swift roaming of client IP addresses through the network to maintain end-to-end connectivity. Existing mobility solutions are mostly based on tunnels, which are inflexible, have high overhead and do not integrate well with the routing process.

The inflexibility in altering the routing and packet forwarding process has been recognized for wired networks. Recently, OpenFlow [12] has emerged as a technology to make switches in wired networks more intelligent and programmable via a standardized interface. The key idea of OpenFlow is to move the forwarding intelligence into a central network control server, while keeping the routers or switches simple. This allows to implement different forwarding approaches efficiently. Moreover, OpenFlow can co-exist with legacy routing protocols and offers the possibility of network virtualization.

In this paper we demonstrate the use of OpenFlow for WMNs. The main contributions are an architecture that allows for the flexible and efficient use of OpenFlow in WMNs and the evaluation of this architecture in a real WMN testbed called KAUMesh. We demonstrate the feasibility of our architecture by implementing a mobility solution within a few lines of code, that enables clients to seamlessly roam between different MAPs without the need for tunneling. To the best of our knowledge there has been no previous work that uses OpenFlow in the context of WMNs and evaluates key performance parameters.

The rest of the paper is outlined as follows: Section II describes the key ideas of OpenFlow and analyzes in detail what the challenges of OpenFlow in WMNs are. In Section III our system architecture is proposed and the implementation is outlined. Section IV evaluates general performance characteristics of OpenFlow in WMNs. In Section V we demonstrate how to use the proposed architecture to handle client mobility. Finally, Section VI compares our approach to related work while Section VII provides a summary and some ideas for future work.

II. OpenFlow and Wireless Mesh Networks

A standard switch/router consists of tightly interlinked elements which handle the forwarding of packets (data plane) as well as the control of the forwarding tables (control plane). This makes switches complex and hard to extend with new functions. OpenFlow addresses this issue by separating the control and data plane. The control plane no longer resides on the switch only, but is implemented partially in a server that runs a network-wide Network Operating System (mostly based on NOX [8]). The data plane is abstracted as a flow-table, which contains a set of rules for flow-processing. In addition, OpenFlow defines a protocol - the secure channel - between the data plane and the Network Operating System.

A rule consists of a pattern, a processing priority, an expiration time, counters and a list of actions. The pattern specifies which packets belong to a flow and can comprise header values or switch ports. Actions allow to manipulate headers and to output the packets to switch ports or encapsulate and forward them to the Network Operating System. When a packet arrives at a switch and no appropriate rule is found, the packet is encapsulated and forwarded to the Network Operating System (usually for all first packets of a flow). The Network Operating System runs applications which decide how the packet should be handled. For example, a rule could be established that all packets belonging to the flow of a given packet should be flooded to all switch ports. As OpenFlow is aware of the flow a packet belongs to and the flows properties, many interesting network services can be implemented efficiently. For example, previous works (e.g. [9], [17]) have shown how to use OpenFlow for multi-casting and network virtualization.

978-1-4577-0638-7 /11/$26.00 ©2011 IEEE

Routing and forwarding in WMNs is similar to routing in traditional wired networks in many respects. However, the characteristics of the wireless channel (e.g. fading, interference, broadcast) lead to a high network dynamicity, which needs to be handled by the routing protocol. WMN routing protocols therefore are often optimized to create little control traffic and react fast on the varying quality of wireless links.

Thus, when using OpenFlow in WMNs the following points should be considered: Due to variations in link qualities and nodes joining and leaving the network, the network topology changes at a much higher pace than in wired networks. Along with the self-configuration requirement of WMNs, this necessitates an autonomous topology discovery and the ability to react swiftly on changes in the network. In addition, as wireless networks do not have the clear notion of a point-to-point link, neighbor and topology discovery need to be adapted to wireless networks. Moreover, the capacity of WMNs is one or two orders of magnitude smaller than the one of wired Ethernets. Hence the overhead introduced by communication between OpenFlow-switches and the network controller deserves special attention. OpenFlow typically uses out-of-band signaling, i.e. the connection to the Network Operating System is on a network separate from the actual data network. To avoid extra hardware costs, the control network and data network can be separate VLANs within one physical network. As the IEEE 802.11 MAC layer does not support VLANs, the 802.11 Service Set IDentifiers (SSIDs) can be used equivalently. Furthermore, the IEEE 802.11 MAC uses ACKs to acknowledge frame receptions. Thus the sender MAC addresses need to be modified as the packet traverses the network. Mesh routers are often based on low-speed embedded device platforms and perform the packet forwarding completely in software. Therefore, the burden of parsing many forwarding rules should be taken into account. Based on these considerations, the next section describes an architecture for OpenFlow in WMNs.

III. OpenFlow in WMNs

A. OpenFlow-Enabled Mesh Routers

In our architecture, the mesh network consists of OpenFlow-enabled mesh routers (see Fig. 1) and mesh gateways. Each node can be equipped with multiple physical wireless cards to implement multi-radio WMNs for increased capacity [11]. Each physical wireless interface is split into two virtual interfaces. One virtual interface is used for control traffic and the other one for data traffic. The virtualization is achieved by the use of different SSIDs. Multi-hop IP connectivity between the virtual control interfaces is enabled using any normal mesh routing protocol. In our case we use OLSR [3]. The data interfaces are connected to the OpenFlow data path. The data path uses local sockets to communicate with the control path component. The control path component connects via the control interface and the secure channel to the NOX, which is located in the core network. Connectivity to outside the mesh is achieved by using one or multiple gateways. In addition, a monitoring agent resides on each mesh node. This agent can be queried to provide monitoring information about link quality, channel utilization etc.

When using multi-radio multi-channel meshes, the initial network setup requires channel assignment (CA) to the radios and links in order to maintain connectivity. A CA-mechanism is out of scope for this paper, but for example the approach in [16] can be used.

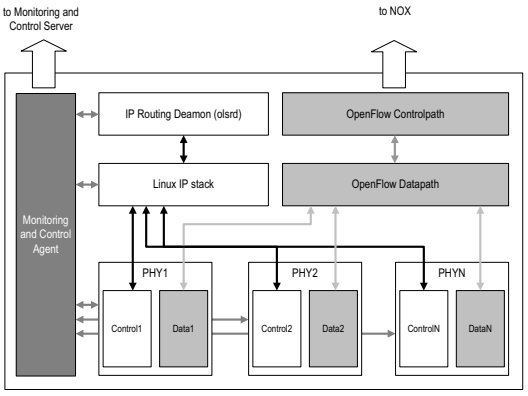


Figure 1: Architecture of an OpenFlow Mesh Node

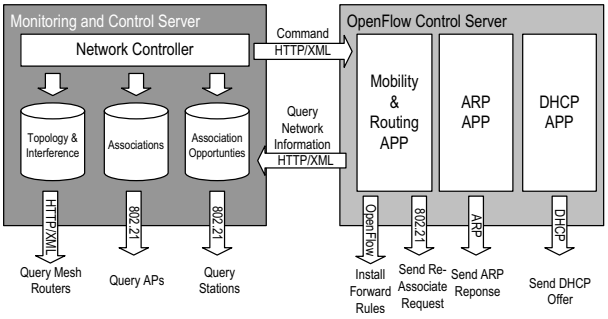


Figure 2: Architecture of the Core Network

B. Core Network

The core network (Fig. 2) comprises a Monitoring and Control Server (MCS) and the NOX. The MCS queries information from the mesh routers and clients and builds a topology, an association and an association opportunity database. Furthermore, a network controller can execute actions, for example to optimize the network operations. The topology database contains a network graph, which is build from the connectivity information of the mesh routing protocol. If a link-state routing protocol such as OLSR is used, all nodes have a view of the whole network. Therefore, the connectivity graph can readily be obtained from e.g. the network gateways. In addition, the network graph can be annotated with link quality metrics, if provided by the routing protocol. The association database contains a list of stations and the MAPs they are associated to and is built from the system statistics of the MAPs. During idle periods, the stations scan the network to find close-by MAPs and report this to the monitoring server, which builds the association opportunity database.

The main purpose of the NOX is to perform routing related tasks, such as setting forward tables, handle node mobility and manage network addresses. To support those tasks, the NOX can access all databases maintained by the MCS. When the topology database changed (e.g. because of a link failure), the MCS informs the NOX, which then updates the routes based on the new network graph. When a station joins the network, the NOX uses the topology database to calculate a path between the station's MAP and a gateway and then installs flow-rules accordingly. The association and association opportunity database can e.g. be used for mobility management and load balancing (see our case-study in Section V).

C. Stations

The stations comply to the IEEE 802.11 standard when associating to a MAP. However, a station might contain a

small monitoring and control agent, which allows the monitoring server and the NOX to query information to build the association opportunity database and to trigger handovers using the IEEE 802.21 [2] command set. Legacy clients without the agent can still associate to the network, but cannot participate actively in the network management, as the NOX cannot trigger actions on such clients, for example to initiate a handover.

D. Implementation

We have implemented the proposed architecture in the KAUMesh testbed [7]. The mesh routers are built upon the Cambria GW2358-4 platform, that uses an Intel XScale IXP435 667MHz CPU. The routers use OpenWRT Backfire (Linux 2.6.31) and the OpenFlow reference implementation for the control and data path [19]. The network virtualization is implemented using the multi-SSID feature of mac80211 and the ATH5K wireless driver [1]. We use NOX 0.5 [8] as the network controller. The communication between the individual components is based on standard protocols. The OpenFlow protocol is used for setting up the flow tables, HTTP/XML for the communication between MCS and NOX and IEEE 802.21 [2] to trigger the handover at the stations. The control network routing is setup using olsrd [3]. The olsr-txtinfo plugin provides the network topology information to the MCS via a Telnet-based interface. The monitoring is performed with Nagios[14]. Furthermore, the mesh nodes and clients contain custom Nagios plugins, which are queried by the Nagios Remote Plugin Executor [14]. The association database is stored in a MySQL database.

IV. Evaluation

In this section, we evaluate the principal feasibility of OpenFlow in WMNs focusing on general performance features of the architecture such as forwarding performance, rule activation time or control traffic overhead.

A. Forwarding Performance

As the data path on the mesh routers is implemented in software, it might become a performance bottleneck. In the current OpenFlow reference implementation, the data path runs in user space, which might create additional problems. To evaluate the achievable performance, we set up a string of three nodes, in which the leftmost node transmits data and the middle node forwards the data to the rightmost node using two separate network cards. The links between the leftmost and the middle and the middle and the rightmost node are tuned to orthogonal channels and use fixed PHY rates of 36 Mbit/s in the 5 GHz band. We made sure that no external traffic disturbed the experiments.

We measured the throughput of a backlogged UDP-stream sent over the two hops. The achievable throughput using normal Linux IP routing without the OpenFlow components was approximately 20 Mbit/s. We repeated the measurements using the OpenFlow data path for packet forwarding. To see the impact of the rule set size between 0 and 100 dummy rules were active and needed to be processed before the actual rule to forward the data is matched. Two kinds of rules were used: simple rules that match by the incoming port and complex rules that also match MAC and IP addresses (and hence need more processing power). As Fig. 3 shows the throughput degrades by about 15% when 100 complex rules need to be processed for each packet. For simple rules the parsing process is very fast and thus does not have impact on

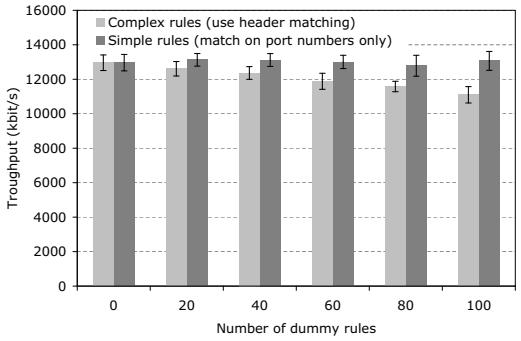


Figure 3: Forwarding performance (Error bars show std. dev.)

the throughput. For routing, IP header matching is required though. Those results show that the present implementation of the data path can indeed create a performance bottleneck on slow mesh-router devices. The results also suggest that keeping the number of rules low is beneficial for performance if user-mode forwarding is activated. A more efficient data path implementation in the OS-kernel might improve performance and is therefore very desirable. Regardless of the size of the rule set, the throughput of the OpenFlow data path was always considerable lower than the normal Linux IP stack.

B. Control Traffic Overhead

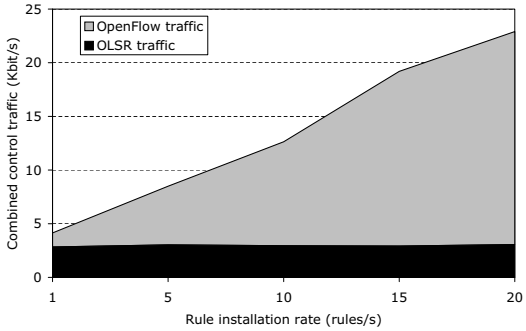


Figure 4: Total control traffic caused by OLSR and OpenFlow

OpenFlow creates control traffic when new rules are installed, statistics are queried and through the heart beat signal that routers send to the network control server, which is located in the fixed part of the network, reachable through gateway nodes. We measured the amount of control traffic that is created when installing rules at different rates at random nodes in the network depicted in Fig. 6. As traffic is relayed over multiple hops it is counted each time it is transmitted wirelessly. In Fig. 4, we compare the OpenFlow control traffic rates to the control traffic created by OLSR, which provides the basic routing infrastructure. As expected, the OpenFlow control traffic increases as the rule installation rate increases, while OLSR traffic stays constant. With 20 new rules per second, the additional control traffic introduced by OpenFlow is about 20 kbit/s and the total control traffic is about 10 times higher compared to a case where only OLSR is used. However, compared to the achievable throughput, the control traffic is still low and for certain scenarios, such as load balancing, much lower rule installation rates can be anticipated. Compared to a pure OLSR network, OpenFlow adds some extra control traffic, but the amount is relatively small.

Scalability is a major concern when using centralized schemes such as OpenFlow. As the network size increases, more heart beat signals are generated and potentially more rules need to be installed. The results from the small test network show that the amount of control traffic generated by each mesh router is in the order of a few kbit/s. As long as the rule installation rate is low, the amount of control traffic should also stay moderate for larger networks. Albeit, scalability issues should be investigated further in later work. For example, by combining the heart beat signals of different nodes in one packet, the control traffic could be reduced.

C. Rule Activation Time

The rule activation time is the duration from when the first packet of a new flow arrives at a node until it is emitted again for forwarding. For new flows, the first packet is encapsulated and sent to the NOX, which then installs a rule on the OpenFlow data path. We measured the rule activation time in the small demo network depicted in Fig. 6. We created new flow arrivals at MAP3 and then correlated packet transmission and arrival times on MAP3 and the NOX to calculate the rule activation time. The PHY rates for this and all following experiments of all links were fixed to 6 Mbit/s.

Fig. 5 plots the rule activation time for different network loads at MAP3. The NOX processing time denotes the time to parse the packet at the NOX and create a new rule. The network delay is the time is takes for the encapsulated packet to travel to the NOX and the time to send the rule to the mesh router. The data forwarding delay denotes the node traversal delay for packets when a rule has been established.

When the background traffic load is low, the rule activation time is smaller than 10 ms. However, as the network load gets high, it takes longer time to transmit the packet to the NOX and therefore the total activation time increases. The processing time at the NOX is in the order of 1.5 ms and might be decreased by a more efficient implementation of the packet-processing application (e.g. by using C++ instead of Python, as in our case) or more powerful server hardware. The network delay for higher loads could be decreased by prioritizing channel access for control packets with IEEE 802.11e. Also, in the present setup the NOX is close to the mesh gateway and therefore the delay in the core network plays a minor role. However, in a real deployment the NOX might be placed in a data-center which is physically and logically far from the mesh network and hence the delay in the core network can be considerably higher.

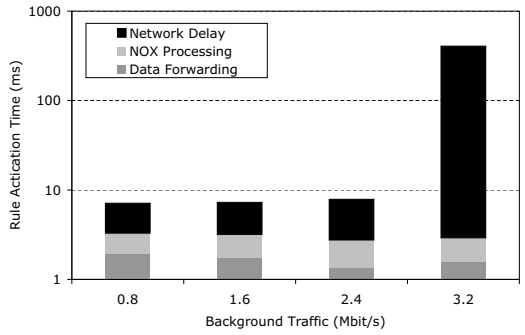


Figure 5: Rule activation time on MAP3 (Avg. of 10 experiments, CoV < 0.001 for all values)

V. Example: Mobility Management with OpenFlow

In order to test the feasibility of using OpenFlow in mesh, we extended our architecture with a simple application for mobility management. To support seamless handovers, a station should keep its IP address even when associating to a different MAP. Handovers are needed to support node mobility, but can also be a method for load balancing: Typically, handover in a WLAN is initiated by the station by scanning for available access points and selecting the one with the highest signal strength. Especially in the mesh context, this might not be a good idea as the access point with the highest signal strength may have poor uplink capacity towards the mesh gateway. This is because in WMNs, typically the path capacity and the uplink capacity of a mesh gateway provide the bottleneck. Here, letting the network decide based on information to what access point to associate may be a better strategy [13]. Therefore, in our architecture, the MCS periodically examines the current client associations. If the MCS decides that a client should need to re-associate to another MAP from its association opportunity database and additional information such as available path capacity, the monitoring server may trigger a handover request at the NOX.

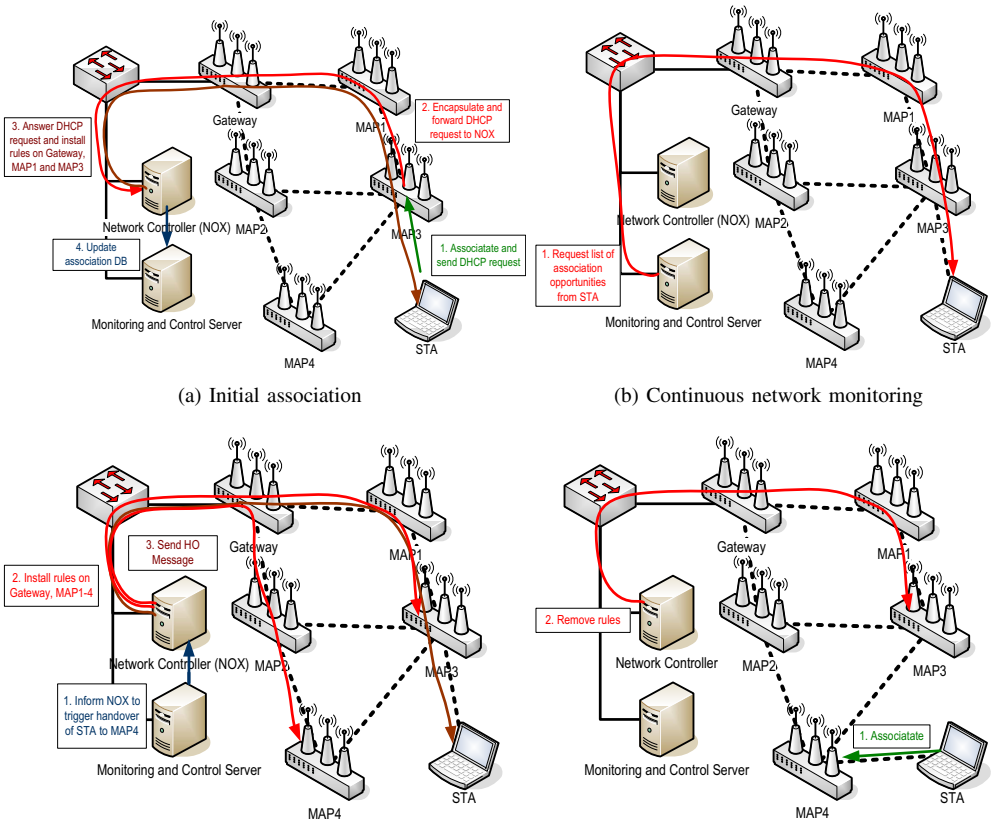
A. Case Study: Mobility Management

The focus of this study is on public Hotspot-like deployments, where typically no link-layer encryption is used and thus key management is not an issue. A network initiated handover, e.g. triggered by the NOX, which has network-wide information on congestion from the monitoring server, is therefore an interesting method for traffic management. The goal of the study is not to provide an optimized solution for mobility management in mesh networks. We were rather interested to show that with a few lines of Python code, a reasonable and useful service can be implemented in our architecture.

In our showcase, the mobility management application sets forwarding rules on the mesh routers to allow stations to access the wired network via the mesh network and the gateways. Furthermore, it offers a web-service interface, which allows to trigger handovers of a station from one MAP to another. In addition, the NOX runs an application to respond to ARP-queries. When a station resolves the IP address of its default gateway, the NOX will answer with the MAC address of the MAP the station is associated to. The clients implement a rudimental IEEE 802.21 Media Independent Handover Function, which is capable to trigger a handover between MAPs upon receiving a *MIH Net HO Commit* message from the NOX.

The procedure to enable client mobility is shown in Fig. 6 and works as follows: First (Fig. 6a), when the station initially connects to the network, it uses the default IEEE 802.11 association procedure to connect to the close-by MAP MAP3. It then issues a DHCP-request, which is forwarded to and answered by the NOX. The NOX registers the client in the association database at the monitoring server. Furthermore, the NOX installs rules on the intermediate mesh routers MAP1, MAP3 and the Gateway, which enable a station to access the Internet via the mesh network.

In the second step (Fig. 6b), the MCS continuously monitors the network. If the needs arises (for example because the capacity of a MAP3 falls below a threshold), the MCS requests a list of potential association opportunities from each STA, which STAs create by passively scanning its environment when no data is transmitted. Then an algorithm (which is outside the scope of this paper) decides which STA should be associated to which AP. If for any STA a handover is required, the monitoring server proceeds the next step.



(a) Initial association

(b) Continuous network monitoring

(c) Handover initiation

(d) Handover completion

Figure 6: Management of station connectivity and mobility

In the third step (Fig. 6c), the MCS informs the NOX to initiate a handover for the station from MAP3 to MAP4. The NOX sets temporary rules, which forward traffic to both MAPs. The NOX invokes the Media Independent Handover Command Function at the station to trigger a handover to the new MAP. The handover request includes MAC and SSID of the new MAP, optionally the channel of the new MAP.

In the final phase (Fig. 6d), the STA disassociates from MAP3 and associates to MAP4. The NOX removes the temporary rules - the traffic is now only forwarded to the STA via the MAP4. The association database is updated and the handover is completed.

B. Handover Performance

In this section, we evaluate the performance of the mobility management application based on the presented approach. It is clear, that for a real deployment, several optimizations are necessary for seamless mobility management. Therefore, we did not compare against several other approaches for mobility management in WMNs. Also, our implementation is still lacking an algorithm, which determines which stations should perform a handover to which MAP at what time. Instead, the handover is triggered by calling the respective function of the Web Service at the NOX manually. Albeit, the performance evaluation shows that our architecture is capable to enable network initiated handovers using OpenFlow in WMNs.

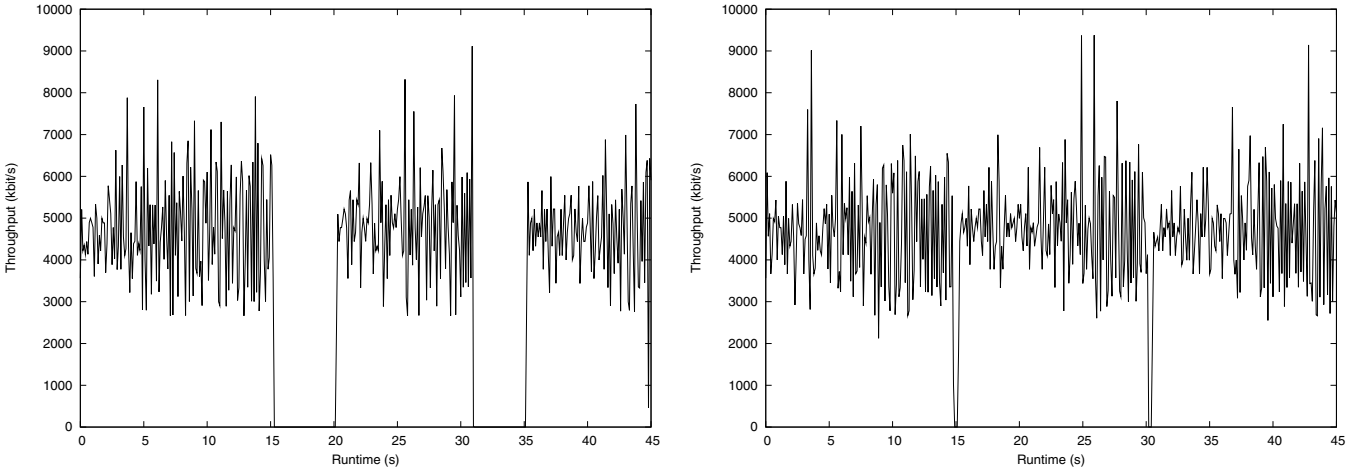
The TCP throughput from a station to the wired network during handovers from MAP3 to MAP4 and back is plotted in Fig. 7. When the handover message does not contain any information about the channel of the new MAP, during a period of almost 5 seconds the TCP throughput drops to 0 (Fig. 7a). During this time the STA scans the network to find the new MAP. However, when the handover message is augmented with the channel of the new MAP, this period reduces to roughly 200 ms. In that case, the STA does not need to scan for the new AP and therefore can perform the handover much faster. As the delay bandwidth products on both paths are small (RTTs ~ 30 ms), TCP recovers fast after the handover.

Table I: Outage duration during a handover

Duration (s)	Without channel information	With channel information
Average	1.87	0.21
Minimum	0.06	0.05
Maximum	2.65	0.27

In the second test, the outage duration during the handover was analyzed closer. Here, a node in the wired network transmitted a UDP datagram every 10 ms to the STA. Table I shows the minimum, maximum and average outage duration (i.e. packet inter-arrival time after handover - 10 ms). The results again show that providing channel information in the handover message to the STA is key to decrease the outage duration. However, even with channel information the outage is on average larger than 200 ms, might lead to severe quality degradations in real-time services such as Voice over IP. The 200 ms are a sum of the dissociation time, the channel switching latency of the wireless NIC and the association time at the new MAP.

During the handover the NOX and intermediate mesh routers are not aware if the station is still associated to the old MAP or already associated to the new one. Hence, the NOX can install rules to forward traffic to both MAPs. Using this temporary packet duplication, the number of packets lost during the handover can be slightly reduced from 25 to 18. Yet,



(a) Handover-message does not include channel of new MAP

(b) Handover-message includes channel of new MAP

Figure 7: TCP throughput from the core-network to the station during two handovers (after 15 and 30 sec.)

the biggest optimization potential lies in a faster association procedure, which is outside the scope of this paper.

VI. Related Work

OpenFlow has been used to provide mobility management before in the OpenRoads deployment [20]. However, OpenRoads does not involve multi-hop wireless communications and integration with existing routing protocols. Different approaches to enable client mobility in WMNs have been proposed. For example, the CARMEN project proposed an architecture for carrier grade mesh networks, which supports mobility by the use of layer-3 tunnels [5]. [4] uses multi-cast trees and tunnels. OLSR-FastSync [18] is an extension to OLSR, which allows fast route-updates after a handover.

The proposed architecture is more flexible than existing ones: It can integrate with other OpenFlow-enabled access and core networks. Furthermore, the architecture can enable new applications, e.g. network access control. Also, the large number of OpenFlow applications currently being developed by the research community should be compatible with our architecture.

VII. Conclusions

In this paper we have proposed a new OpenFlow-based architecture which allows flexible control of packet routing in WMNs. The architecture combines the benefits of OpenFlow (flexible packet forwarding) and WMNs (self-configuration and error-resilience). To demonstrate the usefulness of our approach, we have implemented a simple demo to enable client-mobility in WMNs that allows fast handovers at low complexity and overhead and integration with existing networks. Our testbed measurements confirmed that OpenFlow is an interesting complementary technology to traditional mesh routing protocols and that our simple client mobility solution is feasible for small scale WMN deployments. As future work we intend to develop an algorithm to calculate the optimal STA/MAP associations and flow paths and evaluate the approach in a larger scenario.

References

[1] "Linux wireless." [Online]. Available: http://linuxwireless.org/en/developers/Documentation/mac80211

[2] "IEEE Standard for Local and Metropolitan Area Networks- Part 21: Media Independent Handover," *IEEE Std 802.21-2008*, 2009.

[3] A. Tonnesen and T. Lopatic and H. Gredler and B. Petrovitsch and A. Kaplan and S.O. Tuecke, "olsrd Website," URL: http://www.olsr.org/.

[4] Y. Amir, C. Danilov, M. Hilsdale, R. Musăloiu-Elefteri, and N. Rivera, "Fast handoff for seamless wireless mesh networks," in *Proc. of the 4th international conference on Mobile systems, applications and services*, ser. MobiSys '06. New York, NY, USA: ACM, 2006, pp. 83–95.

[5] A. Banchs, N. Bayer, D. Chieng, A. de la Oliva, B. Gloss, M. Kretschme, S. Murphyk, M. Natkaniec, and F. Zdarsky, "Carmen: Delivering carrier grade services over wireless mesh networks," in *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, 2008, pp. 1 –6.

[6] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626 (Experimental), IETF, Oct. 2003.

[7] P. Dely, "KAUMesh Website," 2011. [Online]. Available: http://www.kau.se/kaumesh

[8] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 105–110, July 2008.

[9] T.-Y. Huang, K.-K. Yap, B. Dodson, M. S. Lam, and N. McKeown, "Phonenet: a phone-to-phone network for group communication within an administrative domain," in *Proc. of the second ACM SIGCOMM workshop MobiHeld '10*. New York, NY, USA: ACM, 2010, pp. 27–32.

[10] D. Johnson, N. Ntlatlapa, and C. Aichele, "Simple pragmatic approach to mesh routing using batman," in *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, CSIR, Pretoria, South Africa, 6-7 October 2008*, 2008.

[11] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *Proc. of the 11th annual international conference on Mobile computing and networking*, ser. MobiCom '05. New York, NY, USA: ACM, 2005, pp. 73–87.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, March 2008.

[13] R. Murty, A. Wolman, J. Padhye, and M. Welsh, "An architecture for extensible wireless LANs," *ACM HotNets-VII*, 2008.

[14] Nagios LCC, "Nagios documentation." [Online]. Available: http://www.nagios.org/documentation/

[15] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561 (Experimental), IETF, Jul. 2003.

[16] K. N. Ramachandran, E. M. Belding, K. C. Almeroth, and M. M. Buddhikot, "Interference-aware channel assignment in multi-radio wireless mesh networks," in *Proc. of INFOCOM 2006*, 2006, pp. 1 –12.

[17] R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T.-Y. Huang, P. Kazemian, M. Kobayashi, J. Naous, S. Seetharaman, D. Underhill, T. Yabe, K.-K. Yap, Y. Yiakoumis, H. Zeng, G. Appenzeller, R. Johari, N. McKeown, and G. Parulkar, "Carving research slices out of your production networks with openflow," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 129–130, January 2010.

[18] S. Speicher, "OLSR-FastSync: Fast Post-Handoff Route Discovery in Wireless Mesh Networks," in *Proc. of IEEE VTC 2006 Fall*, 2006.

[19] The OpenFlow Consortium, "Openflow website." [Online]. Available: http://www.openflowswitch.org/wp/downloads/

[20] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, "Openroads: empowering research in mobile networks," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 125–126, January 2010.