Empowering Software Defined Wireless Networks Through Media Independent Handover Management

Carlos Guimarães[1], Daniel Corujo[2], Rui L. Aguiar[3]
Instituto de Telecomunicações
Universidade de Aveiro, Portugal
Email: {carlos.guimaraes[1],dcorujo[2],ruilaa[3]}@ua.pt

Flávio Silva[4], Pedro Frosi[5]
Faculdade de Computação
Universidade Federal de Uberlândia, Brazil
Email: {flavio[4],frosi[5]}@facom.ufu.br

***Abstract*—Internet access and service utilization has been exploding in mobile devices, through the leverage of WLAN, 3G and now LTE connections. It is this explosion as well that is stressing the underlying fabric of the Internet, and motivating new solutions, such as Software Defined Networking (SDN), to build the controlling support and extension capabilities of the Future Internet. However, SDN has yet to reach the necessary traction to be deployed, and has been more relayed towards experimentation supporting frameworks and away from wireless environments. This paper explores SDN mechanisms and increments them with Media Independent Handover services from the IEEE 802.21 standard, coupling them in a single framework for the dynamic optimized support of OpenFlow path establishment and wireless connectivity establishment. The framework was implemented over open-source software in a physical testbed, with results showing the benefits that this solution brings in terms of performance and signaling overhead, when compared with more basic approaches.**

I. Introduction

The way we have been looking at the Internet has been constantly evolving, motivated by new services, ideas and ways to use it. This plethora of widely different scenarios has been supported by a range of different technologies that simultaneously increment the base functionality of the Internet architecture, and pave the way to even newer enhancements and utilizations. However, those same increments are stressing the Internet's base design, with many core components reaching their limit, hindering further evolutions. As such, a new way to look at the underlying operations of the Internet needs to be done, empowering it towards a new progress and growth cycle.

Software Defined Networking (SDN) appears as a key enabler in this aspect, leveraging a separation from the data plane and the control plane, allowing the later to be run in software and enabling a more horizontal network model. Concretely, different network operations (e.g., routing, forwarding, access control, etc.) can be interfaced by applications via a service-oriented API, allowing the control of the underlying data plane, up to the level of how packets and flows are treated by the network entities. This simplifies infrastructure evolution, avoiding manual configuration, and better adapting to new environments provided by the rise of mechanisms such as cloud computing, the Internet of Things, mobile connectivity and different bandwidth-demanding media-oriented applications.

Since SDN defines a set of mechanisms that rely on fundamental changes on how the Internet operates, it has naturally been met with some resistance at different levels. Deployment initiatives developed by key players, such as manufacturers (e.g., provision of mechanisms embedded in their devices allowing programmable network systems), operators (e.g., virtualizing network services in the cloud and exploration of carrier-enabled SDN) and even datacenters (e.g., Google), have been surfacing. Nevertheless, the main drive has been pushed by research initiatives (e.g., the National Science Foundation Global Environment for Network Innovation (GENI[1]), the OFELIA[2] research project and the New Generation Network Testbed JGN-X[3] in Japan), who have deployed SDN-based mechanisms for operating large-scale experimental frameworks, allowing new protocols to be evaluated in production environments without effecting them. All these utilization initiatives target SDN operations over network control procedures at the core, involving very fast wired links between them.

In order to contribute to SDN integration as a network control and configuration mechanism, we argue that it should be exposed to a greater degree of scenarios, targeting a general deployment as an underlying feature of the base Internet operation. Concretely, considering the growth of wireless-based communications supporting mobile access to Internet services, it becomes important to assess the impact, and contributions, that SDN can provide in these scenarios, when its control plane is able to reach and configure flows right to the network side endpoint of the wireless link (e.g., Access Point, Base Station, etc.), hereinafter referred to as Point of Attachment (PoA).

This is where this paper contributes, by presenting a framework where SDN-based mechanisms, in the form of the OpenFlow protocol, are used to configure the wireless networking nodes and establish communication paths. We explore Media Independent Handover (MIH) procedures, from the IEEE 802.21 standard, to optimize handovers in heterogeneous wireless environments. This approach goes beyond current efforts of integrating OpenFlow with wireless capabilities, that only target the extension of experimental testbed into wireless experimentation. Here, we present how SDN can actually be used to support wireless communication paths, with dynamic mobility management capabilities provided by standards compliant nodes. Results show that this combination allows the provision of enhanced connectivity scenarios, allowing Always Best Connectivity when different handover candidates are

[1]NSF GENI, http://www.geni.net
[2]FP7 OFELIA, http://www.fp7-ofelia.eu
[3]JGN-X, http://www.jgn.nict.go.jp

presented, and triggering OpenFlow procedures preemptively in order to avoid traffic disruption.

The paper is organized as follows. Section II presents the background on SDN and MIH, followed by Section III where our proposed system framework is described. These concepts are evaluated in Section IV, where results of its deployment over a physical wireless testbed are presented, in a mobility scenario. Finally, the paper concludes in Section V.

II. Background

SDN empowers network nodes with the capability of recognizing and applying intelligent behavior to traversing packets, decoupling the control and data planes. This allows the support of scenarios such as large-scale experimentation of new Internet-based protocols over production networks, but also inherently supports more dynamic network topology behavior and (re)configuration. These enhancements are usually demonstrated using OpenFlow [1], a SDN open-source implementation, already available in a number of commercial products and used in several research projects. In OpenFlow Switches, the forwarding operation still resides in the switch, but the control plane runs in software and is able to be managed from a separate network entity, the OpenFlow Controller. This entity configures the switch behavior by modifying the forwarding tables therein, enabling flexible and dynamic network management [2].

Despite these capabilities, some concerns about their software nature were risen, but recent assessments [3] showed that aspects such as scalability are not an issue caused by SDN itself, and can be addressed while maintaining its benefits. Incrementally, with the main scope of SDN action targeting wired connected scenarios, [4] and [5] paved the way to support innovation into wireless mobile networks, exposing OpenFlow software to wireless environments. OpenFlow was implemented as an application on top of OpenWrt[4] [6], allowing commercial wireless routers to act as an OpenFlow-enabled switch. This sparked the study of SDN (and consequently, OpenFlow) under wireless environments, providing a new set of experiment possibilities to be done over supporting testbeds. [7], [8] and [9] study the adoption of OpenFlow in wireless scenarios, focusing on management, monitoring and access control. However, the proposed solutions just consider SDN as an enabler mechanism for wireless protocols experimentation. Moreover, they have some limitations, such as disregarding network resources management, optimization or supporting handover procedures.

The IEEE 802.21 [10] (or Media Independent Handover, MIH) standard defines extensible access media independent mechanisms that facilitate and optimize handovers between heterogeneous networks. It defines a set of Media Independent Commands, Events and Information Elements, made available by a Media Independent Handover Function (MIHF) residing in supporting nodes, which, through the usage of Service Access Points (SAPs) interfacing, abstracts access to information and control procedures of the link layers, independently of their technology (e.g., WLAN, WiMAX, 3GPP, etc.). These mechanisms are used by controlling entities (e.g., high-level mobility management entities) to optimize handover procedures, optimizing connectivity in wireless mobility-supporting scenarios [11]. Although in [7], IEEE 802.21 is referred as a potential enabler to query link information and to trigger handovers in OpenFlow-based Wireless Mesh Networks. Still, no detailed integration of IEEE 802.21 and OpenFlow is addressed, and the potential trade-offs are not even discussed. In particular, IEEE 802.21 is only used for specifying an association request as the handover trigger, providing no considerations on the challenges presented by handovers or mobility management procedures.

[4]OpenWrt - https://openwrt.org/

III. System Framework

The proposed framework integrates OpenFlow network architectures with IEEE 802.21, providing a set of mechanisms that facilitate and optimizes handover procedures. Our framework envisages OpenFlow controlling network flows up to the network side endpoint, affecting, as such, wireless network PoAs (e.g., WLAN Access Point). The media-independent behavior of IEEE 802.21 enables our framework to be technology agnostic, allowing the mechanisms defined here to be deployed over any link technology. Moreover, they also trigger commands for resource availability check, preparation, commit and release, as well as to receive events regarding current link status perceived by the MN. In this way, OpenFlow procedures can use link information to select the best handover candidate (according to both the network and the MN) and to optimize the usage of network resources. As such, our framework enables network management scenarios where, through IEEE 802.21, OpenFlow paths can be dynamically and preemptively configured according to the wireless connectivity opportunities detected by the MN while it moves. With this capability, the OpenFlow path is already established when the MN handovers to the new PoA, reducing the impact to the data sessions involving the MN. Finally, the flexibility of IEEE 802.21 allows it to be integrated with several different mobility protocols, such as PMIPv6 [12], allowing it to accommodate IP mobility procedures when needed.

The proposed framework is depicted in Fig. 1, featuring enhanced versions of the OpenFlow Controller, the OpenFlow Switch and the Mobile Node:

- **OpenFlow Controller / PoS:** This is the heart of the proposed framework, being responsible not only for performing routing related tasks, such as updating forward tables of OpenFlow Switches, but also for handling and controlling mobility procedures. Thus, it couples the functions of the OpenFlow Controller and of the IEEE 802.21's Point of Service (PoS) inside its Mobility Manager module, while featuring a MIHF for exchanging IEEE 802.21 with other nodes.
- **OpenFlow Switch / PoA:** Like the standard OpenFlow switch, this entity is responsible for executing data packet forwarding operations, via a flow table which stores information on how to process each data flow, configured by an external entity (i.e., the OpenFlow controller), via the OpenFlow protocol. It doubles as a PoA, since it provides link connectivity to a MN, as well as features a Mobility Manager module that couples the operations of OpenFlow and

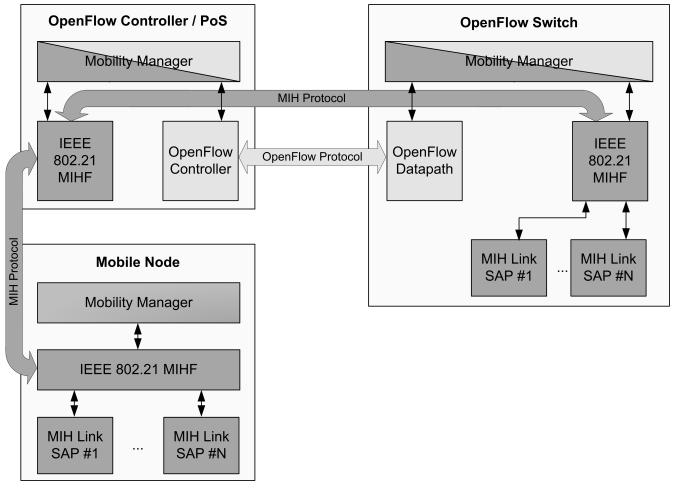


Fig. 1: Proposed framework

IEEE 802.21. Regarding the later, besides the MIHF, it also features a SAP, which allows the Mobility Manager to control aspects of the link interface regarding handover management (i.e., events about link resources establishment), used to optimize OpenFlow procedures.

- **Mobile Node:** The MN represents the end-user equipment that allows the user to connect to the network. The MN may have one of more access technologies, which can be wired (e.g., Ethernet) or wireless (e.g., WLAN or 3G). The MN is coupled with a MIHF, along with interfaces towards the access links (i.e., Link SAPs) and interfaces towards higher-layer entities, allowing them to control and to retrieve information from the links in an abstract way. This interfacing can be done by existing Mobility Managers in the MN or by external network entities that interfaces in a remote way via the MIHF (i.e., the PoS). As such, the MN is able to provide events about detected PoAs or indicating that the current PoA's signal level is decreasing past a predefined threshold, as well as to receive commands to execute handovers to other PoAs (e.g., where OpenFlow has already pre-established flow configuration).

Fig. 2 depicts the proposed signaling for a MN-initiated handover scenario, where the MN changes its PoA due to movement. Although we present a MN-initiated handover scenario motivated by the shifting signal level of the different PoAs while the MN is moving, the flexibility of IEEE 802.21 enables our framework to also support Network-initiated handover scenarios, where, for e.g., the network could detect the overload of a given PoA, triggering the handover of a set of MNs to another PoA. In addition, the link layer abstraction supported by IEEE 802.21 decouples our framework from link related aspects, allowing this scenario to be deployed over different technologies (e.g. Wifi, WiMAX, 3G, etc.) without modifying the signaling presented here.

The handover procedure is initiated when the MN detects a new PoA on its surroundings (1) and, based on several information (i.e., link technology or signal strength), its Mobility Manager decides to handover to the detected PoA. Thus, it triggers a *MIH_MN_HO_Candidate_Query.request* (2) towards its PoS, indicating the detected PoAs that the MN is interested in moving to. Before the PoS replies to the MN, it queries the resources availability on each detected PoA by exchanging *MIH_N2N_HO_Query_Resources* messages (3 and 4) with each PoA. The PoS is then able to provide to the MN an ordered list about the best candidate networks (5) based on other network-based information (e.g., policies). The MN selects the handover candidate and issues a *MIH_MN_HO_Commit.request* message (6), informing the network about an imminent handover to the selected network. Upon the reception of this message, the PoS informs the PoA that a MN is about to move to its network and that any necessary link resources should be prepared by sending a *MIH_N2N_HO_Commit.request* message (7). When the selected PoA acknowledges the PoS request with a successful status (8), the PoS issues an *OFPT_FLOW_MOD* (9) message towards, not only to the PoA, but also to other OpenFlow switches under the domain of the PoS in order to update their forwarding tables. The selection of the OpenFlow switches that require an update of forwarding tables is based on the ongoing sessions of the MN. Thus, the path to the new PoA is established before the handover procedure occurs and, therefore, ongoing sessions can be immediately restored after the handover. Nevertheless, the path to the old PoA is temporarily maintained in order to maintain the ongoing sessions before the handover occurs. In order to receive a notification about the routing update, the *OFPT_FLOW_MOD* message is sent together with a *OFPT_BARRIER_REQUEST* message (10). Thus, upon the reception of the *OFPT_BARRIER_REPLY* message (11), the PoS knows that the routes were already configured and issues a *MIH_MN_HO_Commit.response* message (12) towards the MN, confirming that the resources were

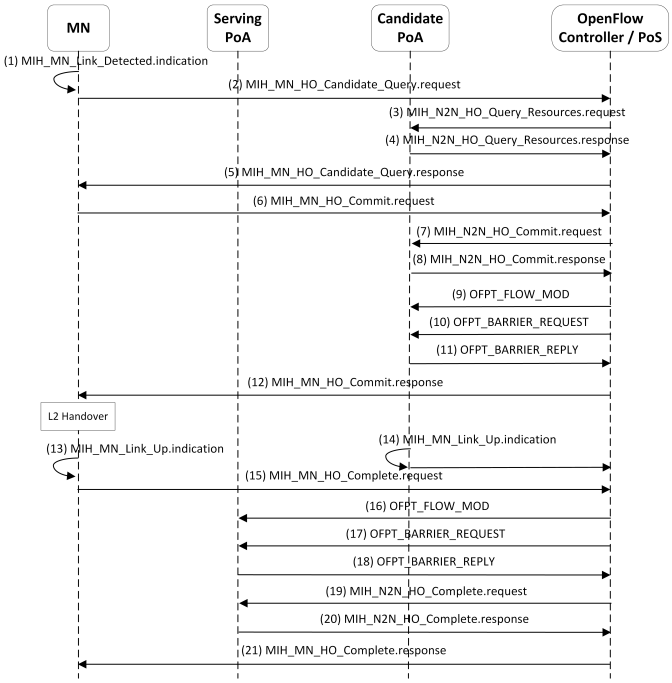


Fig. 2: OpenFlow-enabled IEEE 802.21 handover signaling

successfully prepared by the network. The MN executes the L2 handover procedure informing the PoS about its completion by sending a *MIH_MN_HO_Complete.request* message (15).

In parallel, the new PoA also detects the attachment of the MN, and forwards the event towards its PoS (14). When the PoS receives the *MIH_MN_HO_Complete.request* message, it triggers the OpenFlow procedures to clear routing information related with the MN and the old PoA. For that, it sends a *OFPT_FLOW_MOD* message (16) together with a *OFPT_BARRIER_REQUEST* message (17) to all switches that maintain a route related with the MN and the old PoA. When the confirmation *OFPT_BARRIER_REPLY* message (18) is received, the PoS requests the old PoA to release all resources associated with the MN (*MIH_N2N_HO_Complete* messages (19 and 20)). Finally, the PoS acknowledges the MN that the handover procedure on the network side was completed by sending a *MIH_MN_HO_Complete.response* message (21).

IV. Evaluation

In order to evaluate the feasibility of our framework, we coupled ODTONE[5] [12], an open-source IEEE 802.21 implementation, with OpenFlow 1.3 Software Switch[6] and NOX OpenFlow 1.3 Controller[7] implementations according to the proposals in Section III.

A. Testbed Description

Our evaluation scenario was build over the AMazING [13] wireless network testbed, located on the rooftop of Instituto de Telecomunicações building. As presented in Fig. 3, three different PoAs were selected, each one serving a different IPv6 network, connected to a common switch. The PoAs and switch are OpenFlow and 802.21-enabled. A Content Server, which stores and streams multimedia content, and a MN, which consumes the multimedia content, are the remaining entities that complete the evaluation scenario. This scenario consists on the MN moving through PoA1, PoA2 and PoA3 while receiving a video stream from the Content Server, using the signaling depicted in section III to optimize handover-assisted OpenFlow path establishment. The same network interface card is used between the handovers. Finally, it assumes that the MN is initially attached to PoA1 and already receiving the multimedia content, doing the handover to PoA2 at time 10s and to PoA3 at 20s.

Each physical node is configured with a VIA Eden 1GHz processor with 1GB RAM, a 802.11a/b/g/n Atheros 9K wireless interface, and a Gigabit wired interface. Each node runs the Linux OS (Debian distribution) with kernel version 3.2.0-2-686-pae, with ODTONE and OpenFlow 1.3 Software Switch installed.

Each experiment was run 10 times, showing here averaged results with a 95% T-Student confidence interval. In order to correlate the results, the handovers start exactly at the same time between experiments.

[5]Open Dot Twenty ONE - http://atnog.av.it.pt/odtone

[6]OpenFlow 1.3 Software Switch - https://github.com/CPqD/ofsoftswitch13

[7]NOX OpenFlow 1.3 Controller - https://github.com/CPqD/nox13oflib

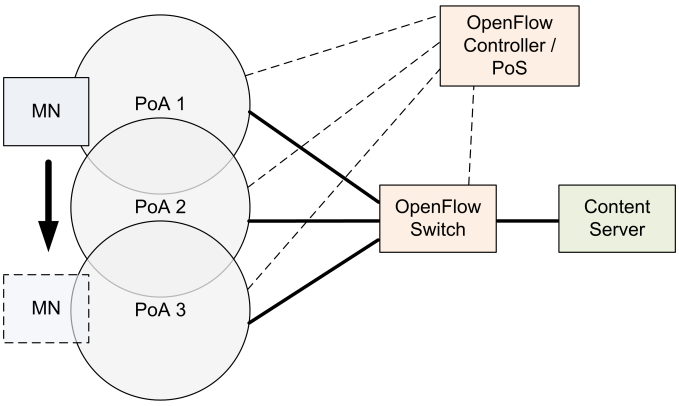


Fig. 3: Scenario description

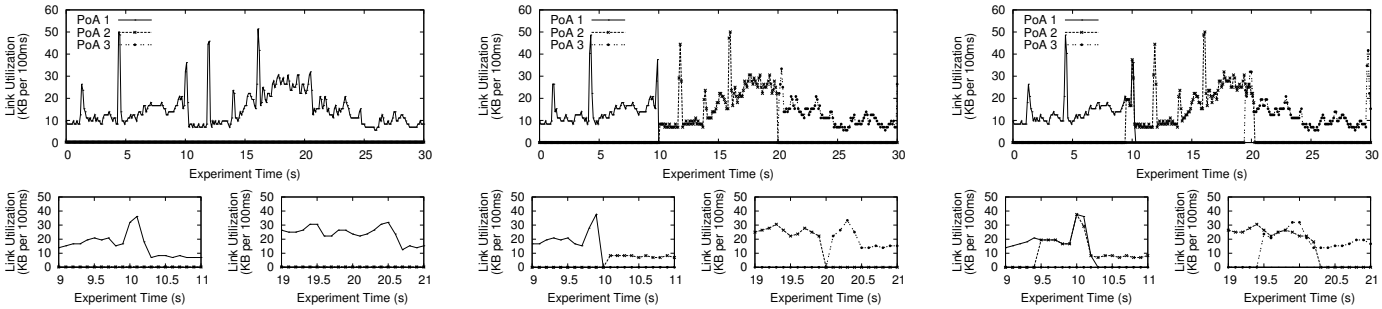
B. Performance Evaluation

In this section, we evaluate the performance of the proposed framework, comparing it with a deployment without any mobility-aware mechanism, and with a deployment featuring some enhanced logic on the OpenFlow controller, where a dummy mobility trigger after the handover initiates OpenFlow procedures. Obtained results are presented in Table I and Fig. 4. Table I shows the impact on the video flow reception, including number of packets sent by the Content Server and received by the MN, percentage of packet lost during the experiment, the delay to restore the video stream after the handover procedure and, lastly, the time during which the video flow was being sent towards a non-required PoA. Fig. 4 depicts the link utilization during the whole experiment, highlighting the time at which the handover occurs (at time 10s and 20s). In order to clearly observe the behavior during the handovers the results of Fig. 4 correspond to a single randomly chosen run.

	No mobility awareness	**Dummy mobility**	**IEEE 802.21 supported mobility**
Total packets sent	3346,8 ± 2.1	3218 ± 10.3	3289,7 ± 24.9
Total packets received	950 ± 15.2	3078,4 ± 21,1	3218 ± 30.5
Total packet lost (%)	28,39 ± 0,44	4,34 ± 0,35	2.18 ± 0,33
Packet lost during HO	∞	70,9 ± 11.6	36,95 ± 12.2
Restore stream delay (ms)	∞	433,9 ± 68.2	197,2 ± 62.2
Redundancy (ms)	0 ± 0	0 ± 0	359,0 ± 62.7

TABLE I: Packet statistics

Analyzing the results of Table I, we can observe that the deployment scenario without any mobility-aware mechanism was not able to recover the video stream after the handover. This happened because neither the MN nor the network were able to detect or report the occurrence of handover and, therefore, the path to the new location of the MN was not configured. In addition, the path via PoA1 (i.e., the initial



(a) No mobility awareness (b) Dummy mobility (c) IEEE 802.21 supported mobility

Fig. 4: Link utilization

point of attachment of the MN) was never removed, resulting in wasted usage of the link resources, as seen in Fig. 4a.

With the dummy mobility triggers, the MN was able to restore the video session after approximately 433,9 $\pm$ 68.2 ms after disconnecting from the old PoA. This delay is related, not only with the L2 handover procedure, but also with the OpenFlow route update procedures which only happens after the handover and its notification to the PoS. From the total of sent packets, the MN lost about 4,34% packets, mostly during the handover procedure. Packet loss is intrinsically related with the delay in restoring the video stream which, as said previously, corresponds to the time required to establish the connectivity with the new PoA and to setup the route towards the new position of the MN. In terms of packets, it represented an average of 70.9 $\pm$ 11.6 packet lost per handover. Looking at the link utilization graphs (Fig. 4b), we can observe that there is no overlap of link utilization by two different PoAs. This behavior occurs because the PoS only establishes the path towards the MN via the new PoA, after releasing the path via the old PoA.

Lastly, from the results of Table I showing the performance of our presented framework, we can observe that the handover performance was significantly improved, where packet loss and stream restoration delay were halved when compared to the previous scenario. The MN was able to restore the video session after approximately 197,2 $\pm$ 62.2 ms after disconnecting from the old PoA. Since the route to the new PoA was already setup up before the handover, after the L2 handover the MN started to received the video stream immediately. Therefore, the time required to restore the video stream is only related with the L2 handover procedure delay. Consequently, it decreased the values of packet loss to about 2.18%, which corresponds to approximately 37 packets lost per handover. However, in contrast to the previous deployment scenarios, the old link is still maintained for about 359,0 $\pm$ 62.7 ms per handover. This time is related with the pre-configuration of the path towards the MN via the new PoA before the handover and the removal of the path towards the MN via the old PoA only after the handover. Fig. 4c shows the link utilization overlap before and after the handover procedure. Of course, this reflects the conservative nature of the chosen handover management algorithm. Our framework is flexible enough to allow the definition of different Mobility Manager strategies, optimizing different aspects, where link utilization overlap can be one of them.

C. Control Signaling Overhead Analysis

In this section we study the footprint of the proposed signaling. For brevity, we focus on the amount of data exchanged and its total time required. The obtained results are presented in Table II, showing the results related with the amount of data exchanged and total time required for different phases of the handover process (i.e., preparation, commit and complete). The values for the amount of data exchanged consider not only the size of the MIH or OpenFlow protocols, but also the size of the L4 headers (UDP or TCP).

		HO Preparation	HO Commit	HO Complete
Size (bytes)	**IEEE 802.21 (UDP + Ack)**	301	294	329
	OpenFlow (TCP)	0	736	688
Time (ms)	**IEEE 802.21 (UDP + Ack)**	11,73 $\pm$ 1.63	128,73 $\pm$ 12.41	122,90 $\pm$ 0.91
	OpenFlow (TCP)	0 $\pm$ 0	56,71 $\pm$ 0,85	56,17 $\pm$ 0.49

TABLE II: Total signaling overhead per handover

Results from Table II show that almost 60% of the exchanged signaling corresponds to the OpenFlow protocol, being the remaining 40% related with IEEE 802.21. The signaling involving the MN accounts for about 20% of the total signaling overhead, due to its participation in the handover process to assist in the candidate query and handover commit and complete steps.

The overhead of the OpenFlow protocol signaling encompasses not only the *OFPT_FLOW_MOD* messages, which update the forwarding tables in the OpenFlow switches, but also the *OFPT_BARRIER_REQUEST* and *OFPT_BARRIER_REPLY* messages, which are used to assure that the routing rules were configured in the OpenFlow switch. A non-verification of the status of the update operation, would reduce in about 144 bytes the signaling related to OpenFlow. In addition, OpenFlow is sent over TCP, introducing the overhead of the TCP header and the corresponding TCP acknowledgment messages. Also, the OpenFlow protocol overhead is

highly dependent on the number of switches on which the PoS should update the forwarding tables. In a very simple way, we verified that the configuration of each OpenFlow switch would increase the OpenFlow signaling between 344 and 392 bytes for the proposed scenario, depending on the flow filter and actions to perform. Lastly, the signaling related with the OpenFlow protocol does not involve the MN, being mainly exchanged between the PoS and the new and old PoAs and the intermediary switch. Regarding time, the OpenFlow procedures took about 112 ms per handover to perform routing update on the OpenFlow switches, divided into 56,71 $\pm$ 0,85 ms to preemptively configure the path through the new PoA and 56,17 $\pm$ 0.49 to release the path through the old PoA.

Regarding the IEEE 802.21 signaling, all signaling involves the PoS, on which about 51% of the exchanged data is related with the communication with the MN, being the remaining 49% exchanged with the PoAs. The old PoA and the new PoA are involved in about 30% and 19%, respectively, of the total IEEE 802.21 signaling. The higher amount of information exchanged with the new PoA highlights its involvement in the resources querying and committing processes, in opposite to the old PoA which is only involved in the resources releasing. The signaling involving the MN is related with its assistance in the candidate query and handover commit and complete procedures. In terms of time, the HO preparation procedures took about 11,73 $\pm$ 1.63 ms, which is minimal compared with the remaining procedures. The HO Commit and HO Complete procedures took about 128,73 $\pm$ 12.41 and 122,90 $\pm$ 0.91 ms respectively. However, besides the time required to prepare and to release the link resources, these values are highly affected by the OpenFlow procedures, since they are encompassed between the IEEE 802.21 procedures.

V. Conclusion and Future Work

The work presented in this article has been framed by the inherent challenges presented to novel mechanisms aiming to optimize the underlying operation of the Internet, such as Software Defined Networking. This work broke from the common utilitarian view typically associated with the deployment of SDN as an experimentation enabler, and to actually contribute to its concrete dissemination as a network control mechanism. In this way, a framework empowering OpenFlow with the Media Independent Handover capabilities of the IEEE 802.21 standard was proposed, allowing an optimized flow establishment and link connectivity, in mobile wireless environments. This allows handover-enhancement processes, provided by configurable indications from wireless link conditions and handover opportunities associated to Mobile Node movement, to dynamically and preemptively trigger software-defined flow configuration, minimizing the impact to on-going data sessions and increasing connectivity opportunities. This framework was implemented over existing open-source software, and deployed in a physical testbed featuring a wireless mobile node receiving a video flow. Results show the benefits that this solution brings in terms of performance and signaling overhead, when compared with more basic approaches. As future work, we are currently evaluating the impact that our framework has in different kinds of information exchanged and flow establishment, considering aspects such as content-centric networking and the Internet of Things, in mobile environments. These place further stringent operation requirements, exposing SDN-based mechanisms to novel scenarios, contributing to its evolution and dissemination.

Acknowledgment

This work has been partially funded by the European Community's Seventh Framework Programme, under grant agreement n. 258365 (OFELIA project) and by the Portuguese Science Foundation (FCT) grant SFRH / BD / 61629 / 2009.

References

[1] 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, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[2] H. Kim and N. Feamster, "Improving network management with software defined networking," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 114–119, 2013.

[3] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 136–141, 2013.

[4] K.-K. Yap, R. Sherwood, M. Kobayashi, T.-Y. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar, "Blueprint for introducing innovation into wireless mobile networks," in *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, ser. VISA '10. New York, NY, USA: ACM, 2010, pp. 25–32. [Online]. Available: http://doi.acm.org/10.1145/1851399.1851404

[5] 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, no. 1, pp. 125–126, Jan. 2010. [Online]. Available: http://doi.acm.org/10.1145/1672308.1672331

[6] Y. Yiakoumis, J. Schulz-Zander, and J. Zhu. (2012, Oct.) Pantou : Openflow 1.0 for openwrt @ONLINE. [Online]. Available: http://www.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT

[7] 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*, 2011, pp. 1–6.

[8] C. Argyropoulos, D. Kalogeras, G. Androulidakis, and V. Maglaris, "Paflomon – a slice aware passive flow monitoring framework for openflow enabled experimental facilities," in *Software Defined Networking (EWSDN), 2012 European Workshop on*, 2012, pp. 97–102.

[9] K.-K. Yap, Y. Yiakoumis, M. Kobayashi, S. Katti, G. Parulkar, and N. McKeown, "Separating authentication, access and accounting: A case study with openwifi," OpenFlow Technical Report 2011-1, 8 2011, made available from July 2011.

[10] LAN/MAN Committee of the IEEE Computer Society, "IEEE Std 802.21-2008, Standards for Local and Metropolitan Area - Part 21: Media Independent Handover Services," 2008.

[11] A. De La Oliva, A. Banchs, I. Soto, T. Melia, and A. Vidal, "An overview of IEEE 802.21: media-independent handover services," *IEEE Wireless Communications*, vol. 15, no. 4, pp. 96–103, 2008.

[12] D. Corujo, C. Guimaraes, B. Santos, and R. Aguiar, "Using an open-source ieee 802.21 implementation for network-based localized mobility management," *Communications Magazine, IEEE*, vol. 49, no. 9, pp. 114 –123, september 2011.

[13] J. P. Barraca, D. Gomes, and R. L. Aguiar, "AMazING - Advanced Mobile wIreless Network playGround," *International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops*, 2010.