Mobility Management and Delay Cost Analysis in SDNized WLAN

Manzoor Ahmed Khan
DAI-Labor, Technische Universität
Berlin, Germany
manzoor-ahmed.khan@dai-labor.de

Patrick Engelhard
DAI-Labor, Technische Universität
Berlin, Germany
patrick.engelhard@dai-labor.de

Tobias Dörsch
GT ARC
Berlin, Germany
tobias.doersch@gt-arc.com

ABSTRACT

The paradigm shift in network control is envisioned as the next evolution milestone of mobile communications. A more flexible control of network technologies will be enabled by implementing the concept of Software Defined Networking (SDN). This dictates that amongst others users' mobility in mobile networks should also be managed by SDN controller. In this paper, we develop necessary tools and software modules to realize the SDNized WLAN. We then implement a distributed mobility management protocol and investigate its performance in terms of handover delays. The paper focuses on studying the delay sources and the impact of system dynamics (e.g., topology size and controller load) on different delay components. To carryout investigations in a more realistic environment, we develop a full scale SDN controlled WLAN testbed.

CCS CONCEPTS

•**Networks** → **Network performance modeling; Network experimentation;** *Network monitoring; Mobile networks;*

KEYWORDS

Software Defined Networking, Mobility Management

1 INTRODUCTION

Recent past has witnessed tremendous evolution of telecommunication networks, which is strongly driven by their positioning as main channel of delivery of most modern services. The principles of SDN does not only enable the centralized control of overall infrastructure but also transform the network to a programmable and flexible network, which can dynamically be shared by the diverse set of network services. However, the popular use of SDN is still confined to Data-center networks, where SDN nodes are interconnected by high speed lines. The problem domain in this case is restrained to optimizing the network behavior only. With SDN concepts blending in the mobile networks, the problem domain widens to include mobility management, which strains SDN's scalability issue even more. The challenges of traditional mobile networks came from the classical design of mobile network, which is hierarchical and mainly centrally controlled. The centralized gateway and backhauling of data consequence in high bandwidth requirements on the bit pipe and single point of failure. Such challenges were addressed by transiting to the flattened architecture. The evolution towards flattened architecture is described in [2], which highlights functional components in the flattened architectures of different network technologies and 3GPP efforts to address the user plane scalability problems.

The architectural evolution has a consistent impact on the way mobility management solutions are implemented. Mobile IP, which was primarily deployed in the hierarchical architecture kept evolving with numerous variants like [3]: HIMPv6, PMIPv6, FMIPv6, Dual Stack MIP, etc. The development of these variants strictly followed the device layer entities and applications requirements (e.g., bandwidth hungry users of 5G, densified networks, and IoT, etc.), which pushed the mobility management concept to a more distributed control approach. IETF just a few years back chartered a working group on Distributed Mobility Management (DMM). This group focuses on stipulating approaches to decompose control and data planes of traditional centrally controlled architecture. Amongst the main goals of DMM, the following were the main highlights: i) Mobility management mechanisms should be distributed and offered dynamically. ii) To reduce the signaling overhead, the mobility mechanisms should be dynamically activated / deactivated. iii) The group should rely on then current mobility management protocols e.g., MIPv6, HMIPv6, PMIPv6, etc.

Distributed mobility management concept may be applied to different segments of the network [3] e.g.; i) DMM in core - realized by distributing the mobility anchors topologically or geographically. ii) DMM in access network - realized by techniques similar to LIPA and SIPTO, which assist in by passing the core infrastructure. DMM allows to decouple the control and data planes, as opposed to classical mobility management approaches, where the control signaling and data pass through a centralized entity (mobility anchor). Hence, separating the data plane, which contributes much bigger portion of the load when compared with control traffic, we to great extent address the scalability issue of mobility anchor. By putting the control plane in centralized node, effective approaches for data plane steering and mobility management be realized. SDN on the other hand is characterized by the similar philosophy i.e., decoupling the control from data plane the SDN controller control the flows on forwarding entities. Owing to obvious complementary characteristics of these two paradigms, literature contributes with a number of proposals for implementing the DMM approaches on SDNized network e.g., [3, 5, 7, 9, 11]. Authors in [9] suggest to separate mobility functionalities of the PMIPv6 nodes and discuss two different possibilities of these functionalities in the SDN architecture i.e.; i) co-location of LMA and MAG at the controller level, ii)

BODYNETS 2016, December 15-16, Turin, Italy

DOI 10.4108/eai.15-12-2016.2267662

LMA located in controller and MAG located in the access switch. A survey on mobility management approaches and their limitations is given in [3]. The authors also discuss potential implementation approaches of the mobility management protocol in SDNized network, which we believe provides a good overview / roadmap of mobility protocols implemented in SDNized controlled mobile networks. [5] highlights the suitability of DMM approaches for 5G networks and proposes the implementation of PMIPv6 in SDNized networks, where the MAG functionality is replaced by a gateway featuring mobility anchoring function and LMA functionality is implemented as control application at the controller level. Similar to (or inspired by) [5], the authors in [11] suggest a DMM solution based on SDN architecture. The authors investigate the handover delays of their contributed mobility management protocol. The implementation of HMIPv6, a variant of MIPv6 on SDNized network is discussed in [7].

Major chunk of the research literature approaches in general discuss the protocols on abstract level by avoiding the implementation details, few do not study the performance in terms of handover delays, whereas the others validate the performance on either emulation platform or numerically analyze them with a number of assumptions. We believe that given the fact that SDNized wireless networks are still in their infancy, deriving conclusions on the assumptions or high level modeling may not support the claims of optimality in mobility management or minimizing the handover delays. In this work, we implement a DMM protocol on a full scale SDNized WLAN testbed. We study the handover delay cost budge on granular level by decomposing it into more granular components and studying the impact of system dynamics over the delay components. This in turn helps understanding the contribution by each delay component to the overall handover delay cost budget. We believe that studying the impact of system dynamics on each delay component to a great extent helps in localizing major source of handover delays and hence the source of bottleneck(s).

2 DISTRIBUTED MOBILITY MANAGEMENT APPROACH IN SDNIZED WIRELESS NETWORKS

In this section, we elaborate on implementation of the PMIPv6 [1] inspired mobility management solution in the SDNized wireless networks. Before we discuss the proposed approach, let us briefly discuss the main objectives of carrying out the control communication between PMIPv6 nodes (i.e., Mobility Anchor Gateway (MAG) ↔ Local Mobility Anchor (LMA)), which may broadly be categorized under the following two objectives: **O1** Mobile Node (MN) Tracking - the MAG should communicate MN's association and disassociation messages to LMA, the central control entity. **O2** Path configuration - for configuration of the network segments on the path between Correspondent Node (CN) and MN.

When it comes to implementing the PMIPv6 on the SDN network, the functionalities of MAG and LMAs may be positioned in different architectural levels e.g., MAG and LMA both encamped at controller level (depicted as configuration 2 in Figure 1) or positioning MAG in a SDN forwarding entity and implementing LMA as SDN native application (depicted as configuration 1 in Figure 1).

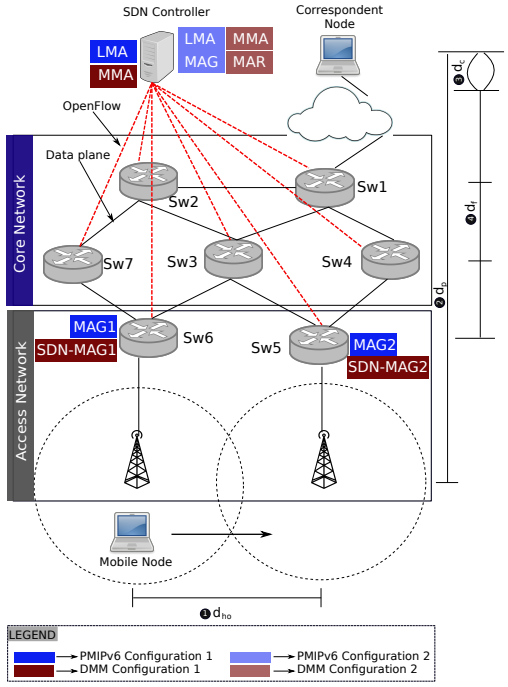


Figure 1: Potential Implementation Scenarios for DMM in SDNized Networks

Figure 1 presents the big picture, which depicts both the topology and protocol architecture.

The proposed architecture implements centralized control plane and distributes data plane. The centralized (SDN) controller implements major mobility management functionalities by Mobility Management Application (MMA) e.g., logic to handle the mobility triggers, path computation and crafting switches configuration logic to be deployed in the switches connecting MN and CN. The edge switch (which interfaces a number of APs on the downstream) acts as an anchor point, which also handles the signaling on behalf of MN. We term the edge switch as SDN-MAG hereafter owing to its adapted functionality for the proposed approach. It is now obvious that the first objective O1 is implemented in the edge switch and the objective O2 is implemented in controller (by MMA). Enabling the control communication between these two entities, we achieve distributed mobility management. Similar to the SDN concept, the controller implements de-facto protocol (OpenFlow) on the southbound to interface the edge switches. Hence, these entities talk the OpenFlow language to each other.

In the following, we provide the sequence and details of interactions carried out between entities for implementing the proposed mobility management approach. Since the call admission control is also implemented by the SDN controller, MN's call request should be extended to the controller over the access and core network entities. When SDN-MAG detects the MN's attachment to an AP interfaced to it, the SDN-MAG translates the MN attachment as the call admission trigger, and maps it to OpenFlow packet-in trigger. The trigger contains MN context information. Upon receiving the trigger, the controller then implements the call admission logic by

registering and computing the flow rules for MN. The flow rules installed on the intermediate switches between MN & CN, which enables the inter-MN-CN communication.

Now if the MN detaches from its current point of attachment, the SDN-MAG translates this in disassociation event (which it receives from the radio part) into OF handover trigger. The trigger is propagated to the controller, which the controller handles by implementing the mobility management application. Upon association to the new point of attachment, the SDN-MAG translates and extend the association event to the controller. The controller computes the new path between CN and MN and crafts the configuration rules. These flow configuration rules are then forwarded through FlowMod messages to all intermediate switches on the new path. The FlowMod basically populates the flow tables of intermediate switches. It should be highlighted that as opposed to the classical implementation of PMIPv6, which uses IP-in-IP tunneling techniques between LMA and MAGs, in the proposed approach the data path is configured by the SDN controller i.e., upon receiving the handover trigger, the controller updates the flow table of intermediate switches toward the new point of attachment. Hence, packets are forwarded without IP tunnel.

3 DELAY COST BUDGET EVALUATION

In this section, we study the delay cost budge of proposed mobility management application in the SDNized WLAN. We know that mobile node will only receive the packets on the newly associated access point of attachment after the handover process is completed. An obvious definition of the mobility cost is; the time between transmission of the last IP packet through the old connection and the first packet through the new connection. Mobility cost budget may be defined in terms of different components, which is strongly influenced by the type of mobility management protocols used, network topology, and network technologies. We decompose the total handover delay cost budget into granular delay components and investigate the impact of different system dynamics (e.g., controller load, topology thickness, etc.) on the overall delay cost budge. Let $\mathcal{D}_t$ represent the total cost budget for mobility management approach, whose delay components are summarized in Table 1. Hence $\mathcal{D}_t$ is given by:

$$\mathcal{D}_t := d_e + d_{p,h}^{e \to c} + d_c + d_{f,h}^{c \to e} \tag{1}$$

4 IMPLEMENTATION OF SDNIZED WLAN

We implement the topology shown in Figure 2. We use commodity laptops connected with a simple hardware switch to create the topology. Mobility application is implemented as native application of OpenDaylight controller. By creating multiple virtual bridges on each physical machine, we are able to create topologies of variable complexity and sizes. To create the topology of variable sizes, we deployed a mix of virtual and physical bridges. To understand the implemented experimental configurations, assume that $\mathcal{L}_{g,o}$ represents topology length, where index g corresponds to two paths $\mathcal{A}$ and $\mathcal{B}$ and o represents the path size (i.e., small, medium, and long). Table 2 summarizes the extensions on each path and total number of hops for the three configurations.

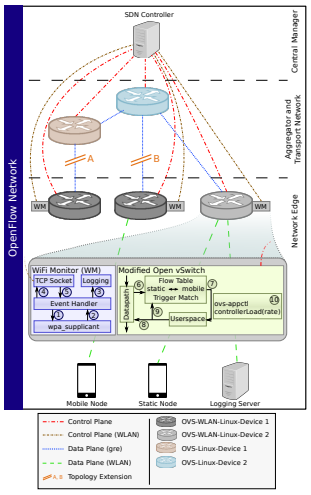


Figure 2: Experimental Setup

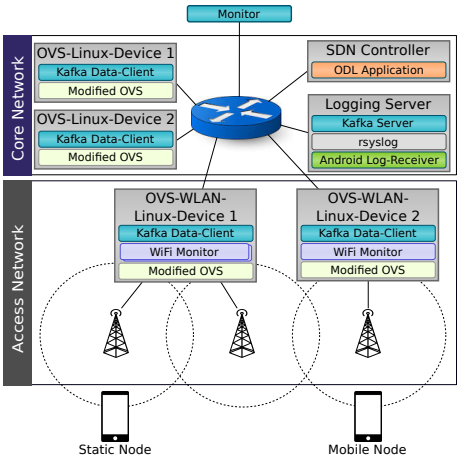


Figure 3: SDNized Wireless LAN - Device Overview

In the following, we briefly discuss our developed tools and software modules to realize the SDNized WLAN. Figure 3 pictorially presents these tools and modules, Table 4 summarizes the inter-tools and inter-entities interactions. Table 3 briefly discusses our developed tools and software modules to realize the SDNized WLAN.

5 DELAY COST BUDGET MEASUREMENT - THE TECHNICAL PERSPECTIVE

In this section, we discuss the technical details of approaches we implemented to capture the delay components mentioned in section 3. The delay cost budget defined by Equation 1 is extended in this section based on the lessons learned from the experiments we carried out in SDNized WLAN. This is to say that the intuitive delay cost budget that we modeled in Equation 1 or the ones presented in research literature with intuitive assumptions may not be realistic representatives of the delay cost budget of handover in the SDNized WLAN. Hence, our extended delay cost budget in

Component	Delay	Remarks
Link Establishment	d_e	Link establishment delay with edge switch (e) i.e., the time required by physical interface to establish new association.
Trigger Propagation	$d_{p,h}^{e \to c}$	MN's (dis/a)ssociation is translated at edge switch (e) and propagated over network hops (h) towards the controller (c).
Computation Delay	d_c	Having received the trigger event, the controller computes the rules to handle the event, which we represent with d_c
Route setup	$d_{f,h}^{c \to e}$	The new rules are transferred and implemented on the switches of newly computed paths.

Table 1: Table Summarizing Delay Components of Mobility Delay Cost Budget

Configuration	Path	Extension	Hop Sum
Small	A	$\mathcal{L}_{S,A} = 0$	4
	B	$\mathcal{L}_{S,B} = 0$	3
Medium	A	$\mathcal{L}_{M,A} = 3$	7
	B	$\mathcal{L}_{M,B} = 5$	8
Large	A	$\mathcal{L}_{L,A} = 6$	10
	B	$\mathcal{L}_{L,B} = 15$	18

Table 2: Topology Overview - Hops

this section captures more realistic cost. Table 5 summarizes the extended set of delay components and their technical implementation details. Based on the delay components detailed in Table 5, we regenerate the Equation 1 in the following.

$$\mathcal{D}_{t,.} = \overbrace{d_{e,a} + d_{e,n}}^{d_e} + d_{p,h}^{e \to c} + \underbrace{\tilde{d_c} + \overline{d_c}}_{d_c} + d_{f,h}^{c \to h} + d_u$$

6 RESULTS AND ANALYSIS

We now evaluate the performance of implemented mobility management solution in authors developed validation framework in terms of delay cost budge. We study the impact of different system dynamics on various aforementioned delay components, of which the most prominent ones include: *i) Impact of topology size on the delay components* - Varying topology size is achieved by deploying virtual and physical bridges in the topology as discussed in section 4 and Table 2. *ii) Impact of control computation load on the delay components* - A brief description on generation of varying controller load is given in Table 3.

We start by analyzing the link establishment delay in Figure 4(a). It should be highlighted however, that this delay component is less relevant to SDN based handovers as it is mainly introduced by hardware and is in theory not affected by topology size or controller load. This assumption is supported by the graphs in Figure 4(a), in which the measured delay is constant and similar independant of topology size and controller load. For WLAN link layer establishment, we have on average experienced delays within the range of $300 - 400msec$.

In Figure 4(b), we study the impact of controller load and topology sizes on the propagation delay component. As obvious from the figure that the varying controller computation load does not impact the propagation delay. However, the change in topology size does have impact on the propagation delay. The propagation delay of the small configuration increased by 70-100% after 4 hops were introduced to the configuration. Similarly further increase of 40-50% is observed when 8 hops are introduced over the the medium configuration. This behavior may be explained by further details of the experiment setup. A very similar behavior is seen for the flow setup delay component, which we see in Figure 4(c). The flow modification messages are sent asynchronously by the controller over the physical network, which is why we expect an increase of this delay component in the larger logical network topology. As these messages are sent to each node on a separate connection, we expect this delay to be higher if there are more OpenFlow nodes in the network to be reconfigured. Since the controller is the node both sending messages and measuring the delay, we expect a minor impact of controller load on this component. This behavior is advocated by Figure 4(c), where the only impacting parameter is topology size.

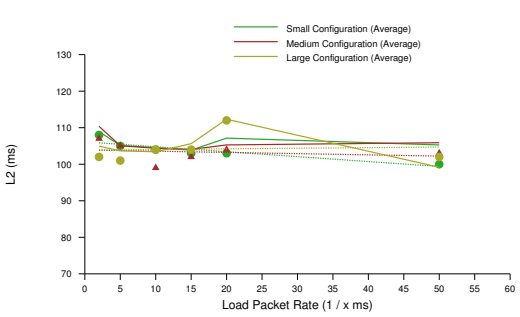
The impact of computation delay is depicted in Figure 4(d). As the core network topology is, in most cases, static, routes through the core network can be cached by the controller. The controller builds the cache the first time it calculates a route through the network, which causes the first computation event of an experiment to be an order of magnitude larger than the following computation delays. As the size of the cache increases with increasing topology size and complexity, we expect to see an increase of this delay component when changing the OpenFlow network topology. The graph shows the computation delay without the initial calculation of the cache. We assume that the lookup-time is negligibly small. Hence, the increase of this delay component stems solely from the higher number of flows that have to be prepared if more nodes have to be reconfigured.

7 CONCLUSION

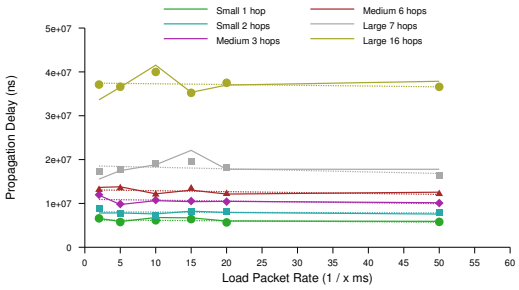
In this paper, we have implemented a distributed mobility management protocol in SDNized network. The performance of the proposed distributed mobility management is evaluated in terms of handover delays. We also studied the impact of controller load and topology thickness on the handover delay values. The experiments were carried out on a full scale SDNized WLAN testbed.

Component	Description
WiFi Monitor	To achieve efficient tracking of mobile hosts, we develop WiFi monitor tool using C++, which runs on linux system. For more please refer to [8]
Monitoring Tool	Keeps track of experiments states and detects anomalies by showing state of active switches. We used Apache Kafka messaging system. It comprises of three components: i) Kafka server - resides in our logging server. ii) Kafka data client - locally monitors switch state. iii) Monitor - contains kafka command client and user interface, where UI displays flow tables of switches.
ODL Application	Build on ODL version 1.4.2-Helium. The application uses the following ODL services: i) Monitoring - contains server to which the WiFi monitor instance is connected. ii) Routing - uses monitoring module to identify the APs associating the communicating hosts. It computes and installs the flows.
Android Application	Functions in two modes namely *Sender* and *Receiver*. The *Sender* sends *UDP_PKT*s to the *Receiver* at a specified rate. The *Receiver* stores a time-stamp, which determines, when a packet was received. After a given timeout, the *Sender* automatically changes to a different Access Point.
Load Generation	For this, we extended the functionality of OVS such that it causes a specified bridge to generate a dummy *PACKET_IN* at a specified rate.
SDN Debugger	For demonstration and debugging purposes, we implement a graphical user interface, which allows users to closely monitor the various components of the experiment. It offers a unified view of all events occurring during the experiment. Since we automate the execution of the experiment, we also implement a replay feature which reads log-files of all nodes recorded in the past as if they were generated in real-time. This enables us to track down odd behavior and errors which occurred during unsupervised experiment runs.

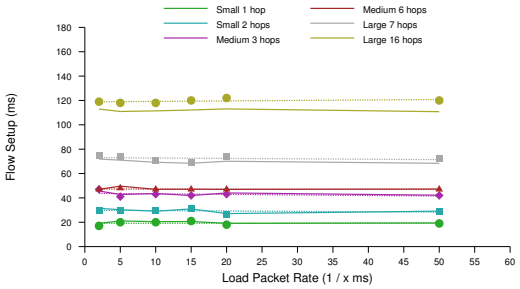
Table 3: Descriptions of components and software modules to realize the WLAN



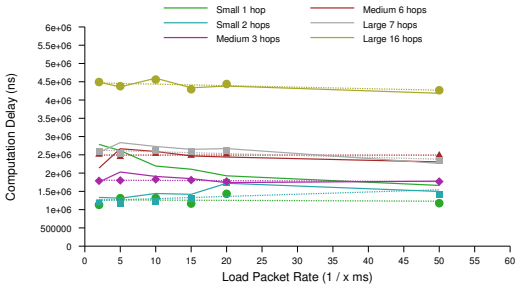
(a) Link Establishment Delay



(b) Impact of controller load and topology size on propagation delay



(c) Impact of controller load and topology size on flow setup delay



(d) Impact of controller load on the computation delay

Figure 4: Experiment results

Notation	Message-Type	Receiver	Sender	Answer	Notes
FMOD	OpenFlow *FLOW_MOD*	OVS	Controller	*FMOD_ACK*	-
PIN	OpenFlow *PACKET_IN*	Controller	OVS	-	Might cause *FMOD*
FMOD_ACK	OpenFlow *PACKET_IN*	Controller	OVS	-	Source MAC 00:00:00:00:00:00; Used for measurements
LOAD	OpenFlow *PACKET_IN*	Controller	OVS	*FMOD*	Source MAC 00:00:00:00:00:01; Loads the controller; *FMOD* executed on dummy OVS
L2_EVT	L2 (Dis-)Association Event	Controller	WiFi Monitor	*L2_EVT_ACK*	Informs controller about host connection
L2_EVT_ACK	Acknowledge of *L2_EVT*	WiFi Monitor	Controller	-	Used for measurements
UDP_PKT	Datagram Packet	Android Receiver	Android Sender	-	Contains running ID and send timestamp

Table 4: Control & Measurement Messages

Component	Sub-comp.	Notation	Remarks
Overall	Sending	$\mathcal{D}_{t,s}$	It is difference in send timestamps s between the last received *UDP_PKT* before a handover and first received *UDP_PKT* after a handover.
	Receiving	$\mathcal{D}_{t,r}$	In this we use the receive timestamps instead of the send timestamps.
Link Establishment	Re-association	$d_{e,a}$	For the measurement of the L2-Reassociation delay, all devices that run an instance of the WiFi-Monitor (i.e. every AP) are synchronized using the NTP protocol ([4, 6]) Every L2-Event is logged to Logging-Server. The difference in timestamps of an L2-Event-Pair (i.e. disconnect-event and corresponding connect-event) is considered to be the L2-Delay. Since NTP server is used for synchronization is within our local network, the difference of clocks of the APs *can maintain synchronization nominally within one millisecond* [10]. L2-Delay is fairly high ($> 100ms$), thus the possible inaccuracy is acceptable for our purpose.
	Connectivity	$d_{e,n}$	As the sending operation is paused during a handover. After the L2-Handover is completed, the Android Application directly tries to continue sending. Time between time to continue sending and the first successfully sent packet is the L2-Connection delay.
Propagation	-	$d_{p,h}^{e\to c}$	It is the time consumed to propagate a (dis-)association event from the Access Point to the Controller. To measure propagation-Delay of an L2-Event, the WiFi-Monitor starts a timer directly before sending a *L2_EVT* message (connect/disconnect) to the ODL-application. The ODL-application answers with a *L2_EVT_ACK* on reception. The previously started timer is stopped as soon as the WiFi-Monitor receives the acknowledge from the controller. This gives us a round-trip-time which is then divided by two to obtain the Propagation-Delay.
Computation	Pre-computation	$\tilde{d_c}$	Time between the reception of a *PIN* on controller and the time when handling of this *PIN* starts. In ODL, each received packet contains a time-stamp defining when this packet was received, which we utilize to determine time passed between the reception and handling this packet.
	Main	$\overline{d_c}$	The time that expires while the controller calculates new flows. It is measured by starting a timer before calculating a route and stopping the timer, right before sending out the corresponding *FMOD*s.
Flow Setup	-	$d_{f,h}^{c\to e}$	The time it takes to apply the previously calculated flow updates. Similar to propagation delay, we measure a round-trip-time by starting a timer on the controller, right before sending a *FMOD* command. We have implemented a functionality in OVS that each switch answers with a *FMOD_ACK* message immediately after the new flow was applied. The timer is stopped as the controller receives the *FMOD_ACK* message.
Unmeasured	-	d_u	The difference between C_A and the sum of all other components. E.g. the ARP resolution, that Android triggers after changing the Access Point.

Table 5: Table illustrating various delay components and their technical details in SDNized WLAN environment

Acknowledgment: This work was partially carried out in a project, iMoveFAN, a collaboration project under Huawei Innovation Research Program. Authors would like to thank Dr. Zoran Despotovic for his feedback and constructive comments.

REFERENCES

[1] Carlos Jesús Bernardos and Gramaglia. 2010. Network-based localized IP mobility management: Proxy mobile IPv6 and current trends in standardization. *JoWUA, Special issue: Advances in Wireless Mobile and Sensor Technologies)* 1, 2/3 (2010), 16–35.

[2] László Bokor, Zoltán Faigl, and Sándor Imre. 2011. *Flat Architectures: Towards Scalable Future Internet Mobility.* Springer Berlin Heidelberg, Berlin, Heidelberg, 35–50.

[3] H Anthony Chan, Hidetoshi Yokota, Jiang Xie, Pierrick Seite, and Dapeng Liu. 2011. Distributed and Dynamic Mobility Management in Mobile Internet: Current Approaches and Issues. *Journal of Communications* 6, 1 (feb 2011). DOI: http://dx.doi.org/10.4304/jcm.6.1.4-15

[4] H. Gerstung, C. Elliott, and B. Haberman. 2010. *Definitions of Managed Objects for Network Time Protocol Version 4 (NTPv4).* RFC 5907. RFC Editor.

[5] F. Giust, L. Cominardi, and C. J. Bernardos. 2015. Distributed mobility management for future 5G networks: overview and analysis of existing approaches. *IEEE Communications Magazine* 53, 1 (January 2015), 142–149. DOI: http://dx.doi.org/10.1109/MCOM.2015.7010527

[6] B Haberman and DL Mills. 2010. RFC 5906: Network Time Protocol version 4: Autokey specification. *Internet Engineering Task Force* (2010).

[7] S. F. Hasan. 2015. A discussion on software-defined handovers in Hierarchical MIPv6 networks. In *Industrial Electronics and Applications (ICIEA), 2015 IEEE 10th Conference on*. 140–144. DOI: http://dx.doi.org/10.1109/ICIEA.2015.7334099

[8] Manzoor Khan, Patrick Engelhard, and Tobias Dörsch. 2016. SDNizing the Wireless LAN - A Practical Approach. In *Proceedings of EUROSIM 2016.*

[9] S. M. Kim and H. Y. Choi. 2014. OpenFlow-based Proxy mobile IPv6 over software defined network (SDN). In *IEEE 11th CCNC*. 119–125. DOI: http://dx.doi.org/10.1109/CCNC.2014.6866558

[10] D. L. Mills. 1985. *Network Time Protocol (NTP).* RFC 958. RFC Editor. http://www.rfc-editor.org/rfc/rfc958.txt http://www.rfc-editor.org/rfc/rfc958.txt.

[11] Tien-Thinh Nguyen, Christian Bonnet, and Jérôme Härri. 2016. SDN-based distributed mobility management for 5G networks. In *WCNC 2016, IEEE Wireless Communications and Networking Conference, Track 3 - Mobile and Wireless Networks, 3-6 April 2016, Doha, Qatar*. Doha, QATAR. DOI: http://dx.doi.org/10.1109/WCNC.2016.7565106