Forging Client Mobility with OpenFlow: an experimental study

Nikos Makris*†, Kostas Choumas*†, Christos Zarafetas*†, Thanasis Korakis*† and Leandros Tassiulas†‡
* Dept. of ECE, University of Thessaly, Volos, Greece
† Informatics & Telematics Institute, CERTH, Greece
‡ Dept. of ECE, Yale Institute for Network Science, USA
{nimakris, kohoumas, hrzarafe, korakis}@uth.gr, leandros.tassiulas@yale.edu

***Abstract*—The wide proliferation of IEEE 802.11 compatible devices and the provisioning of costless Internet connectivity in most cases, have created fertile ground for investigating seamless client mobility and handoff management from a cellular technology to any wireless access point available. Although handoffs and client mobility are currently addressed by the IEEE 802.21 standard along with mobility management protocols such as Mobile IPv6, yet no remarkable efforts exist for the wide deployment of such solutions. Moreover, the adoption of such architectures requires considerable changes in the mobile node's networking stack. In this work, we propose a Software Defined Networking technology inspired scheme for managing client mobility among heterogeneous wireless networks, by adopting changes only on the network edges. Our solution is compatible with the existing IPv4 and IPv6 addressing solutions. By employing the OpenFlow technology on the border of our network with the Internet, we manage to keep both ends of the network aware of any topology changes, and thus preserve any already established connections, resulting in a seamless handoff process. We evaluate our technique in a real network setup, by employing WiFi and LTE technologies and benchmark it using higher layer protocols with multi-homing features, namely Stream Control Transmission Protocol and Multipath TCP.**

I. Introduction

The low cost and rapid deployment of IEEE 802.11 compatible networks has paved the way for the penetration of multi-homed network devices in the market which support multiple interfaces; from smartphones with WiFi interfaces complementary to 3G or 4G cellular ones to smart security systems using WiFi for their primary wireless connection along with a cellular network for backup connectivity. Nonetheless, the mobile operators who provision the cellular networks almost always incur extra costs when using the network for data connections. Although the cost of using such services has significantly decreased over the past years, it is still a notable source of revenue for the mobile operators.

Nevertheless, managing a client's mobility among heterogeneous technologies is not a simple task. In the case of horizontal handoffs, where the mobile user moves among homogeneous networks served by the same authority, preserving the established sessions is managed by a Mobility Management Entity (MME). Examples of such processes in the LTE technology is the utilization of the S1 interfaces between the base stations and the Enhanced Packet Core (EPC) network, or the X2 interfaces for the coordination among the base stations involved in the handoff process.

In the case of vertical network handoffs, the different authorities involved make the uninterrupted handoff procedure a challenge, regarding the established network sessions and the way that will be represented over a different access network. Methods similar to the horizontal handoff process are applicable, like for example in [1] where the authors exploit a scheme for the interconnection of the WiMAX ASN Gateway with the 3GPP UTRAN. However, these methods are solely restricted to a coordination of the access network gateways, which applied in a real world environment will require contracting among the involved network operators. These solutions do not apply for the case of the widely available WiFi networks, commonly found in an urban environment and provisioned even by individuals, which could be exploited for the maximization of the total network capacity.

Therefore, managing vertical handovers (VHOs) in a seamless way, seems to be a real challenge to cope with. To this aim, the IEEE 802.21 standard [2] aims to pave the way for a seamless roaming process among the available access networks. It introduces extensions to the existing network stack, which support seamless connectivity across different Radio Access Networks (RANs). Nevertheless, mobility is managed by higher layer solutions, like Mobile IP or the Session Initiation Protocol (SIP) [3].

In our work, we realize seamless vertical handoffs by managing them only on the network edges, meaning the wireless clients and the destination servers. By exploiting a Software Defined Network (SDN), created using the OpenFlow technology, we are able to accomplish VHOs and keep the ends of the network informed of the changes in the underlying network topology. For our evaluation, we consider the network beyond the access network as a "black box" where we do not apply any changes. We evaluate our technique by collecting performance measurements regarding the handoff process, and compare it against existing multi-homing solutions which can facilitate seamless handoffs, in a real testbed setup.

The rest of the paper is organized as follows: Section II describes the existing protocols, standards and related solutions for supporting VHOs. Section III presents the architecture of our framework. In section IV we present our real testbed setup, while in section V we showcase our experimental findings and benchmarking results of our solution. Finally, in section VI we conclude.

II. Related Work

As multi-RAN access is a key factor to seamless Internet connectivity for mobile users, several protocols have been introduced in order to preserve any established connections as a client roams in areas with dense network coverage. Different protocols exist at different layers of the OSI stack which offer seamless connectivity. One of the most outstanding is the IEEE 802.21 standard, for Media Independent Handovers [4], able to handle connections to/from IEEE 802.3, 802.11, 802.15, 802.16, 3GPP and 3GPP2 networks. A full protocol suite has been defined, for managing subsequent to handoff procedures, like authentication and authorization in the new target network. Nevertheless, since its standardization in 2009, no remarkable deployments of the technology exist. The existing open source solutions, like OpenMIH [5] and ODTONE [6], provide the Service Access Points (SAPs) for the integration of network managers and higher layer mobility solutions [7].

Prior to IEEE 802.21, Mobile IPv4 and Mobile IPv6 have been proposed and standardized with RFCs 4721 [8] and 5944 [9] respectively. Mobile IP (MIP) can constantly identify a mobile client with a permanent IP address, although the client might be using different access networks. Preserving the same address ensures that the connections are not broken in case of a handoff. The MIP protocol defines a home address and a care-of address as follows; the home address is used to associate the user with a home predefined network, while the care-of address is the one identifying the node's current network. In case that the care-of address changes, the mobile node communicates these changes to a home agent (using its home address) and establishes a tunnel connection to the home agent for achieving seamless connectivity to the destination. Open source implementations exist for the MIPv6 protocol, compatible with the recent UNIX system release. For example, *umip* [10] is executed as a user space service for handling and optimizing a handoff, by establishing IPsec tunnels between the home agent entity and the mobile node.

As the available network interfaces on a network node have multiplied, several protocols have been standardized which offer multi-homing capabilities. Although such protocols do not target directly in managing handoffs, they can be efficiently utilized for performing seamless VHOs, even more efficiently than the MIP alternative [11]. The two most outstanding protocols of this category are the Stream Control Transmission Protocol (SCTP) [12] and the Multipath-TCP (MP-TCP) [13] protocol, operating at the transport layer of the OSI stack.

The SCTP protocol has been considered as an alternative to the TCP protocol operation, optimized for wireless technologies. It is aiming at alleviating the Head of Line blocking effect, imposed by the operation of the varying Congestion Window of TCP over a wireless technology, by establishing multiple streams within one association between the two traffic exchanging entities in a network. Nonetheless, SCTP is supporting multi-homing; an SCTP socket connection binds over several network interfaces. One interface is used as the primary and whenever the traffic cannot reach its destination through it, traffic is routed through one of the rest interfaces.

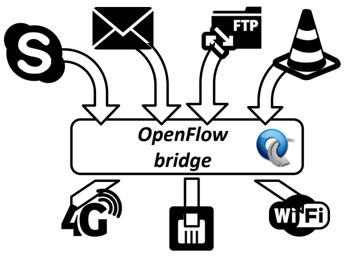


Fig. 1: *The client architecture that we adopt; all the available interfaces are placed under a single bridge instance which all the applications interface*

The backup interfaces are constantly investigated about their connectivity, by means of a heartbeat mechanism, based on a predefined interval. Since the SCTP socket connection is responsible for delivering the traffic to the application layer, the process using the socket remains totally agnostic of any changes, in case of a handoff [14].

Similar to SCTP, MP-TCP can establish multiple connections over multiple interfaces, while the application is presented with a simple TCP connection. Kernel implementations of MP-TCP can ensure backwards-compatibility with the existing protocol stack. MP-TCP can be configured for either establishing multiple connections to an end-point, and thus increasing the total network capacity, or using the rest of the connections as redundant backup links. The backup mode can be efficiently used to provide successful management of handoffs in the environment that we describe [15].

On the other hand, OpenFlow technology has paved the way for innovative usage of the resources in a network, by providing a highly configurable API to the network administrator. Although OpenFlow was initially targeting wired networks, innovative applications can be built leveraging it in wireless networks as well. Authors in [16] argue about the applicability of OpenFlow enabled wireless networks and their capabilities for managing seamless handoffs. In [17] an OpenFlow controller is used to build an experimental testbed for performing handoffs among different wireless and wired technologies. However, the solution relies on complex configurations, leaving out an Internet scale setup, while no indications of the achieved performance are provided. Similarly, [18] and [19] employ the operation of OpenFlow enabled WiFi APs for enabling a seamless roaming experience among them. Nevertheless, all these works are either not applicable in an Internet scale setup, or require controlling the target networks via OpenFlow. In this work, we enable OpenFlow access only on the mobile node and the receiving terminals, thus leaving the operator provided networks completely unchanged and providing a universal solution that is directly pluggable on any contemporary network.

III. OpenFlow based Handoff Control Architecture

The designed framework aims in creating the environment where a mobile client can constantly change the access technology used, while the established connections are preserved.

The applications running on top of our framework shall always remain agnostic to any network changes. To this aim, we employ the OpenFlow technology in order to establish custom flows at the network edges, which alleviate any handoff effects on the already established connections.

During a handoff, network address changes take place at the mobile host, which break the established connections if no proper management is applied. These changes are induced by the different gateway used by each RAN, or by the NAT process that is always present before the traffic is routed to the Internet. Therefore, a proper coordination scheme has to be employed among the involved RAN providers that would enable the client to be discovered using the same address information as was happening before the handoff. Nevertheless, such a scheme would involve contracting among different network providers. With our proposed scheme, the changes are handled at two points; on the client that performs the handoff and just before the traffic reaching the destination server. By using the OpenFlow technology, we are able to establish custom flows on a network switch, by mangling the exchanged traffic accordingly so as the connections are not dropped. From here on, we refer to our framework as OpenFlow Handoff Control (OHC) and we describe the algorithms applied at the mobile client and the destination server of the network.

A. Mobile Client Framework

The key for applying our algorithms relies on creating virtual OpenFlow enabled switches. To this aim, on the mobile node we employ the architecture illustrated in Figure 1; we place all the available networking interfaces in a single switch. By relying on the Open-vSwitch framework [20] for the creation of our switches, the switches residing on the mobile node are OpenFlow enabled. The Operating System on the mobile node interfaces only the bridge device and uses it as the default interface for any outgoing/incoming traffic from the mobile node. The controller that is configuring the flows on this virtual switch is in charge of selecting the appropriate southbound interface (e.g. WiFi, LTE) for exchanging traffic.

The switch device that the node interfaces bears a single IPv4/v6 address in the network, which remains unchanged, and is used as the reference IP address for establishing socket-based connections in the network. Nevertheless, as the mobile user is moving from cell to cell (of even different access technologies), it is quite common that a network DHCP server will assign a new IP address to the network interface of the mobile host. Therefore, since the mobile node shall remain agnostic of any topology changes, the network controller is also in charge of initiating the DHCP process and learning the network addresses which will be used in the new network.

Apart from the information gained by the DHCP process, knowledge about the available next-hop neighbors is required regarding their MAC layer configuration. To this aim, we need to intercept and generate our own ARP protocol messages, in order for the controller on the mobile node to be fully informed of the underlying network topology and present the user with a completely seamless experience. The controller creates ARP messages in the background of its operation and exchanges them with the associated interfaces.

Algorithm 1 OHC - Client Side Flow Control

```
1:  if Inside the Coverage Area of an available Network then
2:      Associate to the Network
3:      Initiate and intercept the DHCP process
4:      Flood all ARP packets
5:      Collect ARP replies of the next-hop destinations
6:      For each received ARP request, send an ARP reply
7:  end if
8:  if Handoff is instructed then
9:      Send Handoff indication to the server side
10:     for all Outgoing Packets do
11:         Change the SRC-IP to the selected intf IP
12:         Change the DST-MAC to the gateway MAC
13:         Change the SRC-MAC to the selected intf MAC
14:     end for
15:     for all Incoming Packets do
16:         Change the DST-IP to the bridge IP
17:         Change the DST-MAC to the bridge MAC
18:         Change the SRC-MAC to the selected intf MAC
19:     end for
20: end if
```

Our goal is to create the illusion to the mobile node applications that the underlying network is interfaced by using a single IP address. The network interfaces (WiFi and LTE) are running their own driver instructing the selection of the network; for WiFi the selected network is the one with the maximum Received Signal Strength (RSS) and the LTE network depending on the network allowed to access defined by the SIM card configuration. As the node is moving across different networks, the low level drivers running on the node's kernel space instruct any potential network changes (e.g. from one Base Station to another). For ensuring that the packets sent over the selected network access technology are properly formed, we apply Algorithm 1. The algorithm ensures that the packets bear the correct IP and MAC addresses for the network that will be used as a gateway, and that the packets which are received over the interfaces are formed correctly before delivering them to the initiating application.

B. Server Side Framework

The respective changes for adopting our framework have to take place before the traffic is delivered to the destination application. Although different processes take place for routing the packet received from each access network (WiFi, LTE), they all normally share a common point; they all route the information based on public IP addresses allocated to each client, or they use a NAT process on the border of their connection with the Internet to translate the clients' IP addresses to other ones (possibly private to public and more applicable to the IPv4 schemes). This fact has to been taken care of at the server side when the mobile node is performing VHOs, as each one of the technologies used is represented by a different set of source IP address and source port at the destination.

Algorithm 2 OHC - Server Side Flow Control

```
1: if Handoff Indication is Received then
2:     Extract SRC MAC, IP and Port
3:     Monitor SRC IP and Port for every incoming packet
4:     if TCP packet then
5:         Monitor packet Seq. No.
6:     end if
7: end if
8: if packet belongs to new flow then
9:     if Packet is TCP then
10:        Monitor Seq. No. && port number
11:        if Seq. No. == Expected Number then
12:            Packet Belongs to Handoff Flow
13:        end if
14:    end if
15: else
16:    if SRC MAC, SRC IP, SRC PORT are equal to the OHC message then
17:        Packet Belongs to Handoff Flow
18:    end if
19: end if
20: if flow is a Handoff flow then
21:    for all Incoming Packets do
22:        Change SRC IP, PORT to the establ. session params
23:    end for
24:    for all Outgoing Packets do
25:        Change DST MAC, IP and PORT matching the handoff flow
26:    end for
27: end if
```

Therefore, in order to cope with the address overloading issue, meaning that the IP address of the mobile host is identifying not only the host but its location as well, we create a simple representation for the mobile host at the receiving end. The process that we adopt for resolving this issue is similar to the policy used at the mobile node; we rewrite the IP addresses and source ports on the packets just before reaching their end point, thus keeping the established sessions uninterrupted, as indicated by Algorithm 2.

Nevertheless, discovery of the new network parameters (address and source port assigned by the NAT process) requires a new approach that will employ inspection of the packets received before establishing a new flow, subsequently to the handoff process. To this aim, we use a simple procedure for identifying the new network parameters; prior to performing the handoff, the mobile node controller communicates this event to the server side controller. This message is a standard TCP message send over the new interface that will be used after the handoff procedure. Upon reception of such an indication message, the server side extracts the information entangled with the new network path (SRC MAC, SRC IP, SRC Port) and starts monitoring the received packets in terms of their source port number and source IP address. In case that the traffic is TCP, the transport layer headers bear a sequence number. By monitoring this number, whenever a packet belonging to an unestablished flow arrives, we can deduce that the flow is a handoff flow. In this case, inspecting only a single packet is enough for extracting the appropriate information for establishing the new flows.

IV. Testbed Setup

For the evaluation of our OHC framework, we choose to implement our solution for managing the handoff process in a real network. We performed all of our experiments in the NITOS indoor wireless testbed, by using the majority of the technologies provided to the experimenters. NITOS [21] is a key testbed in the Future Internet Research and Experimentation (FIRE) initiative in Europe, consisting of 50 outdoor prone to RF-interference and 50 indoor RF-isolated nodes deployed at the premises of University of Thessaly campus building. The rich ecosystem of resources provided by the NITOS testbed is exploited in our setup; we use nodes as wireless Access Points, nodes to connect as wireless clients, one LTE femto-cell and the respective EPC network as our small cell solution and one OpenFlow switch for the network reconfiguration at the server side of our experiments.

The topology that we employ for the experiment setup is depicted in Figure 2. We use one node as a mobile station, with one WiFi and one LTE interface for studying the handoff management mechanism. A second node is used as a WiFi AP, using an Ethernet connection to the experimental OpenFlow enabled network. The installed EPC network, behind the LTE access network is able to route the traffic arriving at the PDN-GW component to this experimental network as well. Finally, a third node is playing the role of the server receiving the traffic stemming from the mobile node. The nodes are high-end PCs, featuring Core i7 processors and 4GBs of RAM, while they are using the 3.14.22-squeezemptcp kernel. The femtocells are commercial ones by ip.access, operating in band 7, with a center frequency of 2655 Mhz (EARFCN equal to 3100), channel bandwidth of 10Mhz and transmission power equal to 15dBm, while we are using the provided services to configure them appropriately [22]. The RF-isolated testbed provides us ideal environment conditions for our experiments; for all of the experiments we logged RSSI and RSRP values for the LTE network equal to -53 dBm and -76 dBm respectively.

On our mobile node we use Open-vSwitch (OvS) for our bridging solution, and enable its control from an OpenFlow controller residing on the same machine. We employed the Trema framework [23] as our solution for implementing our OpenFlow controller. For our experiment setup, we use our controllers to setup the flows for the two different involved datapaths; one on the mobile node (on the OvS bridge) and one on the hardware OpenFlow switch of NITOS, mangling the traffic destined to/from the server side of our experiments.

For the operation of the SCTP protocol, we used the SCTP kernel module, whereas for the MP-TCP protocol the MP-TCP enabled kernel provided by [24]. In order for these solutions to be comparable to the one we propose, we minimize any

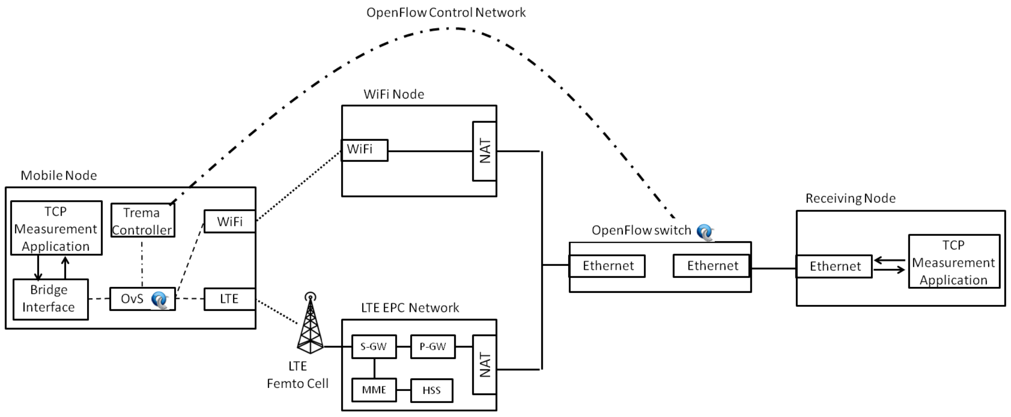


Fig. 2: *The OpenFlow Handoff Control (OHC) architecture that we use to facilitate seamless VHO; the changes that we adopt are only on the borders of the access network, leaving the routing decisions taken at each network provider's site unchanged.*

delay of the network discovery, association and authentication. Therefore, for all the four different solutions that we examine (OHC, MIPv6, SCTP, MP-TCP), we ensure that they are already connected in all the under examination networks.

For the collection of our results, we used the OMF Measurement Library (OML) [25]. OML provides an API to the experimenter for collecting measurement streams from any application which is running over the network. For the collection of our measurements, we extended TCP, UDP and SCTP client/server implementations to collect measurements regarding the delay of consecutively generated messages and total achieved throughput. In the next section we showcase some of our experimental results.

V. Experimental Evaluation

We include two different types of experiments in our evaluation: 1) we initially present some benchmarking results of our OHC framework, in terms of performance overhead and degradation that it incurs to the network operation, since it includes extended packet mangling before delivering the packets to the final recipients (client/server), and 2) we compare it with SCTP and MP-TCP. The nodes that we use are completely RF-isolated and static, so we expect that the rate control algorithms running for each technology allocate the same rate for every experiment. For the experiments using TCP, the congestion algorithm is set to *Cubic*. The experiments using MP-TCP employ the *fullmesh* path manager, as our experiments have showed that it outperforms other available solutions in the MP-TCP enabled kernel [24] like the *binder* algorithm in case of consecutive handoffs. Finally, for the case of the SCTP experiments, we present experiments using different heartbeat intervals, namely 500 msecs and 1500msecs. Setting the SCTP heartbeat interval does not incur any better network performance for our setup, since SCTP exploits the same network buffers for the data and control packets that it sends.

A. Benchmarking of OHC

For the first set of experiments, we measure the overhead in the handoff process imposed by the OHC rewriting functions. To this aim, we initially measure the achieved throughput performance for the different access technologies involved. Since the maximum achieved throughput of the under investigation wireless technologies requires less processing power for handling the incoming packets, we choose to engage a Gigabit wired link for our benchmarking as well.

The results measuring the achieved throughput with and without the OHC scheme, which mangles the packets at the two network edges, are depicted in table I. The results clearly indicate that for the rest of our experiments, the OHC scheme induces no overhead for the network operation, since the achieved throughput for all the cases is almost the same as if no packet mangling was taking place.

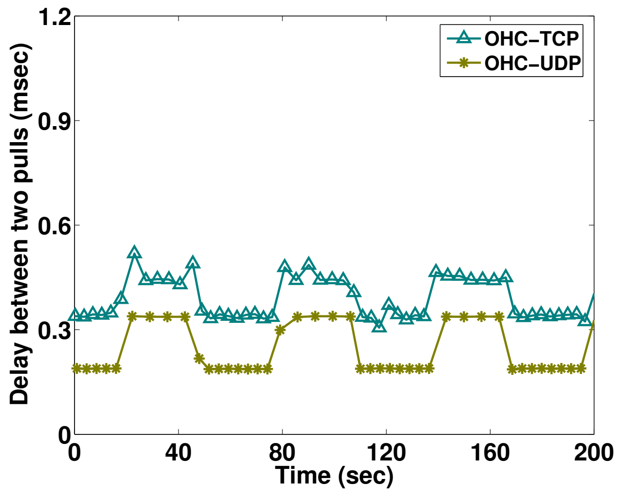
TABLE I: *Throughput achieved with/without OHC*

Technology	**Throughput with OHC**	**Throughput without OHC**
Ethernet	941,44 Mbps	941,45 Mbps
WiFi	24,08 Mbps	24,08 Mbps
LTE	19,57 Mbps	19,55 Mbps

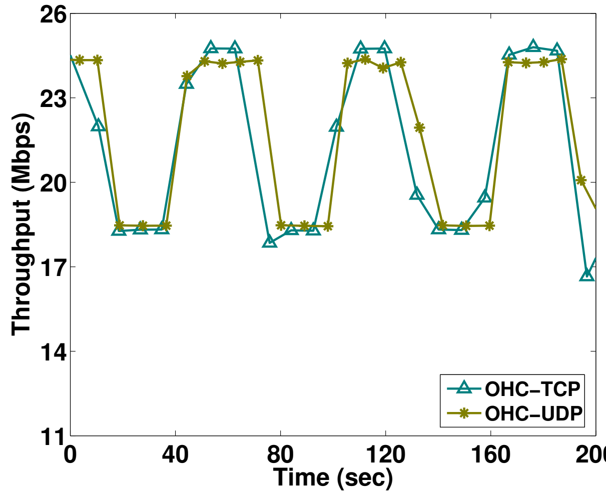
A second set of benchmarking results can seen in figure 3a. We focus on the throughput and delay of consecutive messages, when using our scheme to send TCP and UDP traffic. The messages sent have size of 1420 bytes and are generated constantly at the mobile node. Handoffs are instructed to happen every 30 seconds. For the first value measured and depicted on the diagram, the mobile node uses WiFi and at 30 seconds switches to LTE. As we can see, TCP generates packets with a larger interval of approximately 0.3 msecs when using WiFi technology and 0.45 msecs when using LTE. On the other hand, UDP has delays of approximately 0.2 msecs when using WiFi and 0.3 msecs for LTE. These measurements result from the non-blocking nature of the UDP socket implementation, contrary to the TCP one. Although UDP sends more packets over the selected network, the achieved throughput is similar for both protocols. This is illustrated by the total number of datagrams lost for these UDP experiments (63,27%), as shown in table II.

B. Evaluation against other solutions

For the second set of experiments, we are employing the LTE equipment available in NITOS. Every experiment that

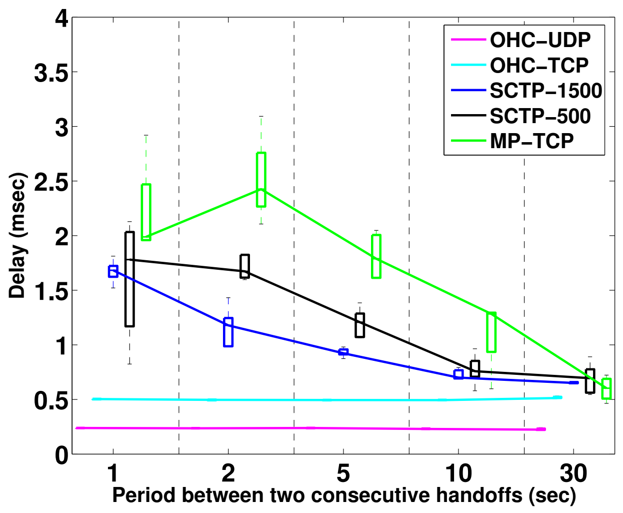


(a) Delay of consecutive messages for OHC. Handoffs happen every 30 secs.

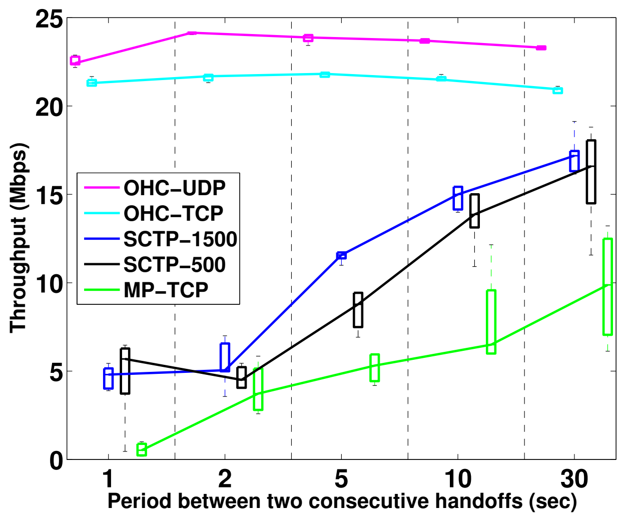


(b) Throughput achieved by OHC. Handoffs happen every 30 secs.

Fig. 3: Benchmarking of OHC framework with different types of traffic.



(a) Average handoff delay for the different under evaluation protocols for handoffs every 1, 2, 5, 10, 20, 30 secs.



(b) Average throughput measured for the different under evaluation protocols for handoffs every 1, 2, 5, 10, 20, 30 secs.

Fig. 4: *Evaluation of our framework against other solutions.*

we are conducting are again with a resolution of 10 times. In Figure 4 we present the indicative results for SCTP with heartbeat interval set to 500 msecs and 1500 msecs, and MP-TCP, when executing experiments with a handoff decision. In Figure 4a, we can clearly see that TCP is adapting faster compared to SCTP to the new network topology. The packet generation delay of SCTP is higher but constant for any network used, compared to TCP and OHC framework. This relies on the implementation of the SCTP sockets in the UNIX kernel, which seem to be outperformed by the TCP ones. On the other hand, although MP-TCP is able to achieve almost the same delays with TCP when no handoff happens, in the case where we have a topology change, its adaptation mechanisms in order to determine the new path are not fast enough.

These facts have an impact on the throughput achieved by each technology (Figure 4b). As we can see, OHC can adapt faster to topology changes, due to the synchronization of the OpenFlow entities, and achieves higher total throughput for each technology used. The SCTP higher packet generation delays are illustrated in the total throughput that we measure for it. Although the total throughput is lower than TCP, we can observe small drops when a handoff takes place. Moreover, setting the heartbeat interval is a tradeoff between delay and total throughput achieved; lower heartbeat intervals are translated in lower handoff delays but consume part of the network capacity. Regarding the MP-TCP performance, we observe that initially, and whenever no handoff happens, it performs similarly to TCP. However, if a handoff decision is taken, the path determination algorithm that it is using is adapting very slowly to topology changes. This is illustrated by the delay for consecutive message generation which is abnormally high, compared to the rest of the solutions.

We evaluate the protocols and frameworks for handoffs happening every 1, 2, 5, 10 and 30 secs. As we can see in Figure 4b, the average throughput that is accomplished when using our scheme is reaching the total capacity of the available UL channels of our topology. However, in the cases when handoffs from one technology to the other happen every 1 seconds, UDP is able to perform better since it generates new datagrams as long the new connection is established. The congestion control algorithm of TCP does not manage to adapt well to these rapid changes and therefore the total throughput achieved is less compared to UDP.

TABLE II: *UDP packet loss for the OHC measurements*

Handoff Interval	**Packet Loss**
1 sec	62,43%
2 secs	62,94%
5 secs	57,99 %
10 secs	62,93 %
20 secs	60,77 %
30 secs	63,27 %

However, as the handoff interval changes, we see that for the cases of 5 seconds and more, the performance of the two protocols converges over our framework. The average delay for the generation of messages for the two different types of protocols seems to converge as well in the cases where the handoff decisions are taken with an interval of 2 or more seconds. In the cases when handoffs happen very rapidly (per 1 sec), the average delay for packet generation in the TCP case is higher, since our algorithm determines the new return path for the pending ACKs.

VI. CONCLUSION

In this work, we presented our OHC scheme, which is directly plug-able to today's networks and can manage efficiently wireless handoffs by employing OpenFlow technology at the network edges, while the end user enjoys a completely seamless experience. We evaluated our solution for a different set of configurations, and compared it with other solutions for managing mobility directly or via multi-homing functions. The results we received are encouraging in terms of performance, since our solution outperforms the existing solutions.

As future work we foresee the application of our scheme on Android based smartphones for enabling fast handoffs among the available interfaces. This operation is likely to create performance issues, given the limited processing capabilities of smartphone devices.

Another future extension of our work is the creation of the local SAPs inside the controllers for the unification of our solution with the ODTONE implementation for IEEE 802.21. Although existing works unify ODTONE with the Linux Network Manager [7], they are able to receive messages of an imminent handoff but not able to address the network addressing issues that OHC solution does.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant agreement no 612050 (FLEX Project).

REFERENCES

[1] W. Song, J.M. Chung, D. Lee, Ch. Lim, S. Choi, and T. Yeoum. Improvements to seamless vertical handover between mobile WiMAX and 3GPP UTRAN through the evolved packet core. *IEEE Communications Magazine*, 2009.

[2] K. Taniuchi, Y. Ohba, V. Fajardo, S. Das, M. Tauil, Y.-H. Cheng, A. Dutta, D. Baker, M. Yajnik, and D. Famolari. IEEE 802.21: Media independent handover: Features, applicability, and realization. *Communications Magazine, IEEE*, 2009.

[3] G.P. Silvana and H. Schulzrinne. SIP and 802.21 for Service Mobility and Pro-active Authentication. In *Proceedings of the 6th Annual Communication Networks and Services Research Conference*, 2008.

[4] IEEE Standard for Local and Metropolitan Area Networks - Part 21: Media Independent Handover Services, IEEE Std. 802.21, 2008.

[5] Y. Lopez and E. Robert. OpenMIH, an Open-Source Media-Independent Handover Implementation and Its Application to Proactive pre-Authentication. In *Proceedings of the First International ICST Conference on Mobile Networks and Management (MONAMI)*, 2009.

[6] Daniel Corujo, Carlos Guimaraes, Bruno Santos, and Rui L. Aguiar. Using an open-source IEEE 802.21 implementation for network-based localized mobility management. *IEEE Communications Magazine*, 2011.

[7] A. Pires, D. Corujo, and D. Gomes. EMICOM: Enhanced Media Independent COnnection Manager. In *Proceedings of sobre Redes de Comunicaes*, 2012.

[8] RFC 5944: IP Mobility Support for IPv4, Revised, 2010.

[9] RFC 3755: Mobility Support in IPv6, 2004.

[10] UMIP mobile IPv6 and NEMO basic support implementation for Linux, available from http:/umip.org.

[11] F.Y. Leu. A novel network mobility handoff scheme using SIP and SCTP for multimedia applications. *Journal of Network and Computer Applications*, 2009.

[12] RFC 4960: Stream Control Transmission Protocol, 2007.

[13] RFC 6824: TCP Extensions for Multipath Operation with Multiple Addresses, 2013.

[14] F. Siddiqui and S. Zeadally. SCTP multihoming support for handoffs across heterogeneous networks. In *Proceedings of the 4th annual Communication Networks and Services Research Conference*, 2006.

[15] Ch. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. Exploring Mobile/WiFi Handover with Multipath TCP. In *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks*, 2012.

[16] Hao Yu. Software-Defined Networking in Heterogeneous Radio Access Networks. In *Proceedings of 30th TERENA Networking Conference (TNC)*, 2014.

[17] R. Izard, A. Hodges, J. Liu, J. Martin, K.C. Wang, and K. Xu. An OpenFlow Testbed for the Evaluation of Vertical Handover Decision Algorithms in Heterogeneous Wireless Networks. In *Proceedings of the 9th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2014.

[18] A. Ardiansyah, P. Paramitha, M. Salman, and D. Choi. Implementation and Performance Analysis of Mobile Handoff Process on OpenFlow-based Wi-Fi Network. *International Journal of Technology*, 2015.

[19] W.S. Kim, S.H. Chung, C.W. Ahn, and M.R. Do. Seamless Handoff and Performance Anomaly Reduction Schemes Based on OpenFlow Access Points. In *Proceedings of the 28th International Conference on Advanced Information Networking and Applications Workshops*.

[20] Open-vSwitch, An Open virtual switch, Available from http://openvswitch.org.

[21] NITOS: Network Implementation Testbed Laboratory using Open Source platforms, Available from http://nitlab.inf.uth.gr/NITlab.

[22] N. Makris, Ch. Zarafetas, S. Kechagias, T. Korakis, I. Seskar, and L. Tassiulas. Enabling open access to LTE network components; the NITOS testbed paradigm. In *Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015.

[23] Trema: Full-Stack OpenFlow Framework in Ruby and C, Available from http://trema.github.io/trema/.

[24] C. Paasch, S. Barre, et al., Multipath TCP in the Linux Kernel, available from http://www.multipath-tcp.org.

[25] M. Singh, M. Ott, I. Seskar, and P. Kamat. ORBIT Measurements framework and library (OML): motivations, implementation and features. In *Proceedings of the 1st International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2005.