Testbed Innovations for Experimenting with Wired and Wireless Software Defined Networks

Kostas Choumas[1], Nikos Makris[1], Thanasis Korakis[2], Leandros Tassiulas[3], Max Ott[4]
[1] University of Thessaly and CERTH, Greece
[2] Polytechnic Institute of New York University, United States
[3] Yale University, United States
[4] NICTA, Australia
e-mail: kohoumas, nimakris@uth.gr, korakis@poly.edu, leandros.tassiulas@yale.edu, Max.Ott@nicta.com.au

***Abstract*—Widely available and remotely accessible testbeds have been used for a direct comparison of innovative protocols and ideas with existing technologies. Therefore, multiple testbeds have been established, aiming at providing experimentation services with both wireless and wired networks. In this context, several frameworks have been developed that enable easy experimentation with the heterogeneous resources that the testbeds provide. However, most of these testbeds aim only in wireless/wired networking experimentation, resulting in unsimilar testbed control and experimentation tools. Several attempts have been made towards bridging this gap in order to allow experimentation with heterogeneous wired and wireless resources. In this article, we present our contributions in extending the state-of-the-art control and management framework for wireless testbeds with support for Software Defined Networking resources. As a proof of concept, we demonstrate two use cases that take advantage of our extensions using novel architectures and present our findings.**

I. Introduction

Software Defined Networking (SDN) is an approach in networking that decouples the process of taking forwarding decisions (control plane) from the underlying system that forwards the traffic (data plane). Over the last years, several theoretical underpinnings on SDN have been established, thus inspiring the conception and deployment of many novel network architectures. Most of these architectures feature individual but unified backbone and access network segments, where the former ones rely on high performance wired connections and the latter ones exploit the benefits of the wireless medium. The wired connections provide high reliability and availability, while the wireless ones leverage on easily deployed access networks. Subsequently, numerous initiatives [1], [2], [3] have emerged which fund projects aiming at bridging wireless and wired networking and building enhanced end-to-end systems with the use of SDN technologies. These projects are abstractly divided in two classes. The first class focuses on theory-driven protocol and architecture design, which is expected to boost the flexibility as well as the utilization of the existing network infrastructures, while the second class targets at the implementation and evaluation of each proposed design under real experimentation platforms.

The experimental platforms provided by these initiatives are essential for the research community, since consensus is growing that a complete and detailed theoretical study of a network problem has to be followed by an applied extensive validation of the corresponding results. As a consequence, research in joint wired and wireless SDN requires testing platforms equipped with the appropriate resources that enable realistic and large-scale experimentation. For instance, the OpenLab [1] and FIBRE [2] projects developed and enhanced heterogeneous testbeds, such as NITOS (wireless networking), PlanetLab-Europe (wired networking), w-iLab.t (wireless networking), i2Cat (SDN) and University of Bristol (SDN and Optical networking) islands. Similarly, the SmartFIRE [4] project aims in building a unified SDN enabled testbed between Europe and South Korea. These testbeds enable experimentation on both wireless and wired network topologies by making use of virtual or physical OpenFlow [5] switches. OpenFlow is a protocol able to separate the control and forwarding plane of the switches, being the most widely known SDN enabler.

Another key issue in the consistent operation of these testbeds is their efficient and flexible management. A framework that enables the control and management of their resources, as well as the description and execution of experiments with use of these resources, is very essential. The experimentation on both OpenLab and FIBRE environments is based on the state-of-the-art cOntrol and Management Framework (OMF) [6], enabling user-friendly scripting and conduction of experiments. OMF is a widely adopted framework by the Future Internet Research and Experimentation (FIRE) [3] community in Europe and the Global Experimentation in Network Innovation (GENI) [7] in the United States. It is open source and integrates all the necessary functionalities about the life-cycle of experimentation on testbeds. The founders and main developers of OMF initially focused on the administration of wireless resources and then moved to a more generic approach where wired links among wireless nodes are also handled.

This article outlines our development efforts in extending OMF to support SDN capabilities. Our goal is to illustrate the feasibility of experimentation in SDN using wireless and wired network topologies, by providing platforms where various scenarios can be tested. The open source nature of the adopted

framework enables further extension and modification of the developed contributions. To sum up, in the following Section II we illustrate the former status in testbed experimentation before our contributions. In Section III, we present the SDN capabilities supported by integrating our extensions in the newest version of the experimentation framework. Implementation of use cases that take advantage of these capabilities is covered in Section IV.We conclude in Section V.

TABLE I
COMPARISON OF THE FRAMEWORKS.

Testbed Frameworks	Experimentation control		Federated Resource Provisioning
	Wireless support	Wired support	
OMF [6]	yes	yes	yes
OCF [8]	no	yes	yes
ProtoGENI [7]	limited	yes	yes
NEPI [3]	yes	yes	no
Teagle [3]	no	no	yes
TEFIS	no	no	yes
FEDERICA	no	no	yes

II. STATUS IN TESTBED EXPERIMENTATION

Large scale experimentation has been enabled through the establishment of several testbed islands over the past years. They feature multiple heterogeneous resources, remotely available and widely utilized, leveraging on cutting-edge networking equipment. However, the efficient operation of a testbed is strongly related to the availability of an easily accessible interface that enables sufficient provisioning and management of its resources. The existence of such an interface is one of the most attractive testbed features for their users, especially when this interface enables the one-click execution of experiments described in an easy-learned and human-readable language. cOntrol and Management Framework (OMF), is fulfilling successfully these prerequisites, thus boosting its wide adoption by several testbed operators worldwide. Several other smaller-scale frameworks exist, applied only in wireless/wired testbeds exclusively.

OFELIA Control Framework (OCF) [8] is a widely used tool enabling experimentation exclusively on top of OpenFlow compatible wired networks. Nevertheless, OCF does not support the control of wireless devices that might be present in one single testbed site. Similar to OCF, ProtoGENI [7] is another widely used control framework that focuses on the cluster experimentation, operating virtual machines that are hosted on powerful servers. Nonetheless, it lacks the flexibility of adapting in heterogeneous testbed environments, since its ability to control multiple kinds of resources is rather limited. In the same context, multiple research projects make remarkable efforts in solving multiple open issues on testbed provisioning, experimentation and federation. However, their use from only a limited number of testbeds does not guarantee their off-the-shelf adoption and smooth operation. An example of this case is the NEPI [3] framework. As far as NEPI is concerned, it leverages on the standardized interfaces of OMF, and provides the experimenter an alternative python based interface for handling the testbed resources. Other frameworks target only in providing a federated resource discovery, reservation and provisioning (like TEFIS, FEDERICA, Teagle [3] etc.) and no experiment control. Table I provides an overview of a comparison between the aforementioned frameworks.

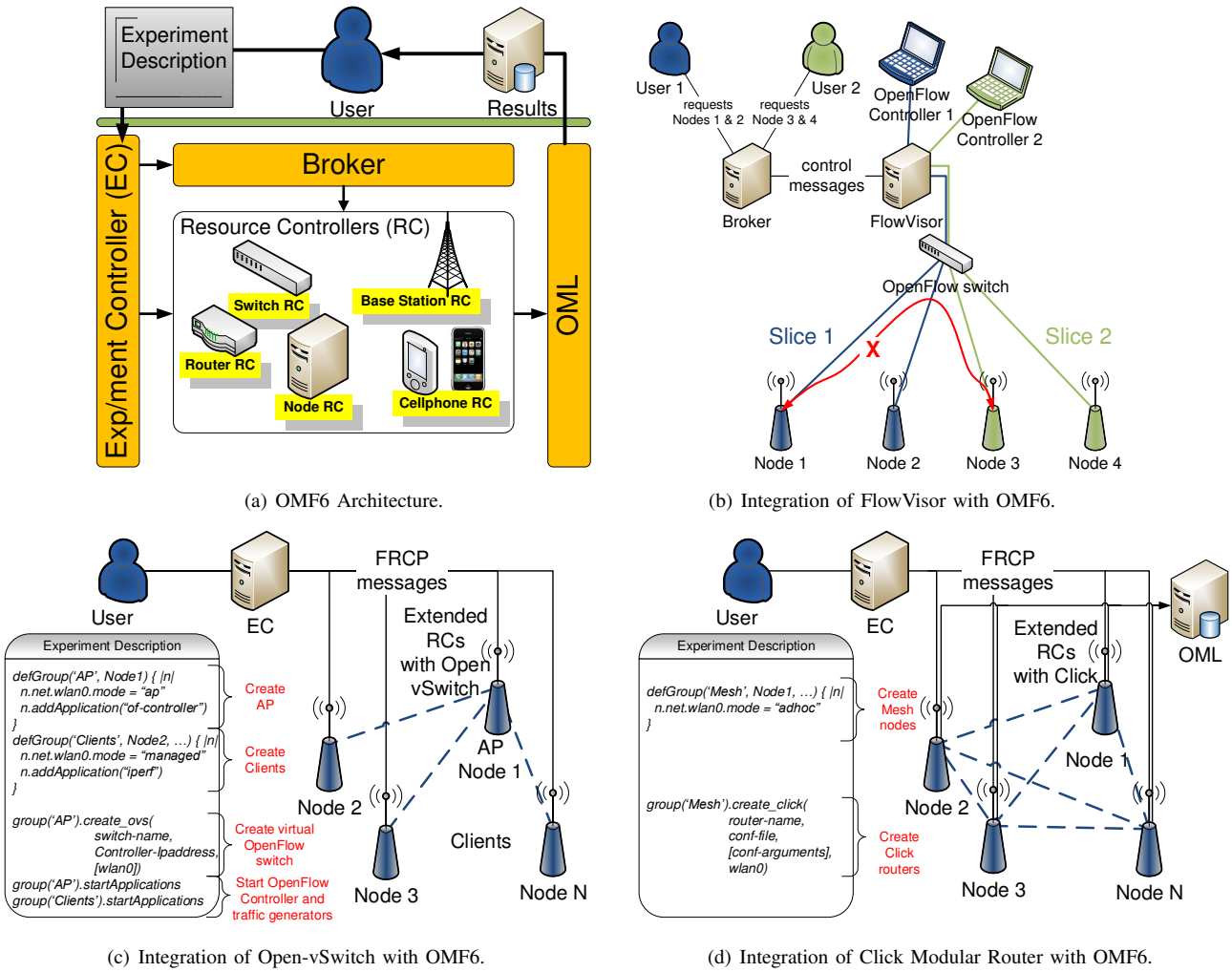
OMF is a generic framework; stemming from the wireless world, its open source and easily extensible nature has been a vehicle for its wide adoption among many testbed operators, regardless of the type of resources they provision. OMF is written in a the Ruby scripting language, with a plethora of libraries available for network specific operations. Its architecture is modular (as illustrated in Figure 1(a)) consisting of different components endowed with the operation of the experiment orchestration and the resource control. Using a simple human readable experiment definition, OMF is supporting the whole experiment lifecycle, cooperating also with its accompanying framework, OMF Measurement Library (OML). The experimenter submits a simple script to the OMF Experiment Controller (EC) and the underlying functionality is responsible for setting up the resources, running the defined applications and collecting the results in an organized way. We can easily conclude that due to its flexibility and mild learning curve, OMF has boosted real world experimentation in testbed facilities.

The newest version of OMF, named OMF6, is even more extensible enabling the enhancement of its core functionality supported by the Broker [9] and the adaptation to various testbed specifications. More specifically, OMF6 defines a standardized protocol that may be used for controlling resources of any type, such as computers or sensors, networking equipment or any other SDN component. This protocol uses a standardized sequence of messages sent by the EC to the Resource Controllers (RCs) and vice versa. The RC is a daemon that behaves as a proxy between the EC and the resource, translating the messages of the EC to executable commands for the resource and vice versa. Testbed operators are able to use this flexible protocol to extend the experimenter's control on new testbed resources or even establish federations of testbeds, thus enhancing the experimentation ecosystem. In the following section we present our work, based on the OMF6 framework, that facilitates wireless and wired SDN experiments in a real testbed environment.

III. BRIDGING WIRELESS AND WIRED SDN EXPERIMENTATION

The research community that carries out the majority of the SDN based efforts has been mainly focusing on the wired However, wireless connections are improved in terms of throughput efficiency and reliability, while they still offer low-cost connectivity for end-devices. The analysis and evaluation of the performance of various end-to-end scenarios, that usually involve a wireless access network and a wired backbone infrastructure, is one of the most attractive research fields [10]. Towards satisfying this growing demand for experimentation, testbed operators have been extending their platforms appropriately, with either SDN-capable, wired or wireless resources.

In this context, we choose to adopt the state-of-the-art OMF6 testbed framework and extend it for handling the



(a) OMF6 Architecture.

(b) Integration of FlowVisor with OMF6.

(c) Integration of Open-vSwitch with OMF6.

(d) Integration of Click Modular Router with OMF6.

Fig. 1. Architecture of the described extensions

respective SDN components. In the following subsections we present our extensions to integrate the *FlowVisor*, *Open-vSwitch* and *Click Modular Router* functionalities in the OMF6 platform.

A. Testbed OpenFlow slicing

As the demand for testbed experimentation increases, the efficient utilization of the testbed resources is one of the main goals for each testbed operator. Slicing is assisting towards this goal, especially for large-scale facilities with subsets of resources that lie idle otherwise. OMF6 satisfies these demands out-of-the-box, taking advantage of its slicing capabilities. The provisioning of isolated experimental slices is facilitated by the component named *Broker*. Our developed slicing service for OpenFlow switches is offered complementary to the traditional slicing scheme imposed by OMF6. Slicing on such a switch is usually performed using the *FlowVisor* [11] framework (Figure 1(b)). FlowVisor behaves as a network hypervisor, which enables the concurrent usage of an OpenFlow switch by multiple experimenters.

In particular, each OpenFlow switch uses a predefined protocol in order to communicate with a remote server. This server is called OpenFlow controller and is endowed with the process of communicating software defined flows to the switch, regarding the switching decisions that it shall take upon arrival of an unknown network frame. FlowVisor is nothing more than a special purpose OpenFlow controller, which acts as a transparent proxy between any OpenFlow switch and multiple experimentation specific OpenFlow controllers. From the perspective of the OpenFlow switch, FlowVisor is its controller. It isolates parts of the underlying hardware switch and provides access to these subparts to experimentation specific controllers. Slicing might depend on several attributes of the switch, like for example the number of ports used, the physical switch memory or processing power utilized per controller instance. The slicing may also be based on the packet flow characteristics, like the IP/MAC source and destination addresses or the VLAN tagging.

We extended the Broker entity to control the FlowVisor process and allocate completely isolated OpenFlow switch

slices upon a user's request. The slices are isolated based on the switch's physical ports, thus preventing each experimenter to interact with the traffic intended for another slice. When a user reserves testbed nodes attached to a physical switch, the Broker transparently creates an OpenFlow slice, consisting of the ports where the reserved nodes are attached. As it is illustrated in Figure 1(b), with the existing Broker functionality, each user reserves two nodes that share a wireless connection using an idle or non interfering with other users' wireless frequency. Our extensions take place at the wired OpenFlow enabled backbone connection of the nodes, and upon the node reservation set up the appropriate FlowVisor instance which abstracts the testbed switch that the two depicted nodes connect to.

B. Virtual OpenFlow switches

The creation of virtual OpenFlow switches relies on the *Open-vSwitch* [12] tool, which is used in multiple commercial products and runs in many large scale production networks. Open-vSwitch is a software tool for creating SDN based networks, using computers instead of dedicated network devices. An example of the wide Open-vSwitch adoption is the PlanetLab-Europe testbed, which upgraded its functionality by supporting OpenFlow capabilities, thus enhancing the testbed's initial usage and upgrading it to one of the largest scale SDN experimentation facilities. With the use of this software, experimenters are able to create an overlay OpenFlow network between the PlanetLab-Europe nodes, that now play the role of virtual OpenFlow switches.

Although Open-vSwitch has been initially developed for managing wired networks by creating network bridges, it can be efficiently used for managing wireless interfaces that are parts of a bridge. If such an interface is placed in an Open-vSwitch bridge, the experimenter has the ability to intercept the traffic that is exchanged over the wireless interface as Ethernet based frames (since the wireless header is removed upon each packet reception by the wireless driver). Although this seems to be a time saving advantage for the researcher, it also poses many questions regarding the controllability of the SDN enabled wireless switch. To this aim, we enable an SNMP agent process on the wireless nodes, which allows us to remotely configure the wireless interfaces in a software defined manner.

Based on these processes for creating wireless OpenFlow switches, we developed the corresponding extensions to the OMF6 framework that allow this functionality. We significantly extended the *Node RCs*, which are in charge of receiving the proper configuration messages and applying the corresponding settings to the underlying physical machines (nodes) of the testbed. All the existing commands of the Open-vSwitch API are supported by our extended RC. Complementary to this, we developed the appropriate exchange messages among the OMF6 entities for instructing the RC to send the appropriate *snmp-set* commands for configuring the wireless interfaces, and the *snmp-get* commands for retrieving their status. Accordingly, the EC entity, which is in charge of sending the appropriate messages to the RC based in the experiment description submitted by the user, has been extended to support this functionality.

The messages exchanged are based on the *Federated Resource Control Protocol (FRCP)* standardized by the OMF6 research community. With our extensions, the experimenter can now use the testbed framework to transparently create and configure virtual switches, combining even wireless resources, in large scale using a user friendly and human readable experiment description. An example of such a set up is the configuration of an Open-vSwitch instance consisting of the wireless interface of a node, the execution of an OpenFlow contoller to control this switch, and multiple wireless clients to connect and generate traffic accordingly on the wireless network in a single experiment definition. Figure 1(c) illustrates the ease of experimentation on this scenario using OMF6.

The aforementioned OMF6 extensions have been developed and evaluated using virtual switches combining several heterogeneous wireless technologies that are available at the NITOS testbed. Namely, our extensions support the concurrent operation and configuration of Atheros and Intel based WiFi interfaces, Intel and Teltonika WiMAX interfaces and Huawei LTE interfaces in a single OpenFlow wireless switch.

C. Software Defined Routers

Click (Modular Router) [13] is another long established software tool that its capabilities can be exploited for SDN development. More specifically, Click enables the development of software defined routers with use of Linux operated physical machines. In Roofnet [14], Click developers investigated wireless connectivity issues and proposed a routing algorithm named after it. Their framework is extensible and well documented, enabling the implementation of many routing algorithms with significantly low effort. The alternative option for packet forwarding in a wireless mesh is the 802.11s protocol, that relies on a similar approach called path selection. Nonetheless, Click is much more flexible and extensible than the existing 802.11s implementations.

Our extensions to support the Click framework have not been so straightforward as for the other two frameworks. Click is a highly configurable tool with many users being able to develop their own extensions of the supported functionality, using the so-called *elements*. As it is illustrated in Figure 1(d), we follow a different approach in order to support as many as possible configurations. The Node RC is only responsible for executing the Click router in the user-space level with the appropriate arguments. The experimenter submits to the EC a configuration file that describes the desired Click settings. With this approach, the experimenters can now define new elements, which did not exist at the time that our developments took place, and use them to orchestrate their experimentation in large scale mesh networks. We have moved one step beyond in the extension of our framework and enabled OML support in the core Click system, responsible for capturing the output of Click execution and injecting the measurements in a database. Using our provided hooks in the Click version 2.0.1, the

experimenter can easily support new measurements from the latest released elements.

IV. Experimental Scenarios with use of Wireless and Wired SDN

In this section we present some experimental results that clearly illustrate the potential of the experimentation in joint wireless and wired SDN. We illustrate the significance of our contributions by running realistic use cases over our enhanced platforms. In particular, we present two separate scenarios, where we try to cope with various research challenges that require deep investigation and experimentation for defining the best approaches.

A. Building content-based private networks in geographically expanded areas

The objective of this scenario is the implementation of large-scale network infrastructures that combine a wireless access network and a wired backbone. The extended use of Virtual Private Networks (VPNs), able to define broadcast-domains with expanded coverage area all over the world, is a strong motivation for improving their capabilities. Their efficiency is boosted significantly when they rely on schemes that balance the traffic load among multiple traffic flows. Experimenting over such these facilities seems to be very attractive in scenarios involving multiple geographically distributed users who request common contents. In such cases, directing the users' requests to a set of content servers, rather than overloading a single end-point, is the optimal solution. The capacity and performance of these networks can be significantly improved, when alleviating the traditional addressing schemes and imposing a content-based approach. A content-based approach with load balancing capabilities can be efficiently implemented by using SDN techniques, like the OpenFlow technology.

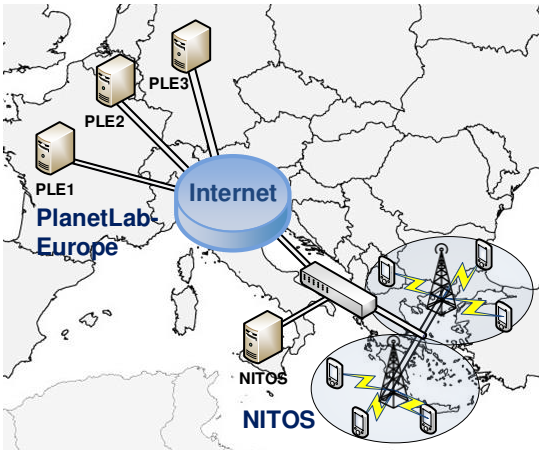
An enhanced large-scale network which supports multiple opportunistic end-point connections and load balancing capabilities, should be based on wireless access-points that are interconnected with a set of content servers through an "intelligent" wired backbone. In [10], authors blend SDN-capable wireless and wired resources in a single pan-European topology. The experimentation on this scheme is based on OMF6, utilizing the Open-vSwitch and SNMP related add-ons. In [10], a VPN is used among wireless access-points/stations, geographically distributed content servers and virtual/physical OpenFlow switches. More specifically, the functionality of the broadcast domain imposed by the VPN is modified, by altering the operation of the Address Resolution Protocol (ARP) and implementing load balancing schemes. The modified broadcast domain connects multiple European-wide sites provided by the OpenLab platform, including multiple content servers from PlanetLab-Europe (PLE1, PLE2 and PLE3) and several wireless nodes from NITOS (including smartphones and the NITOS machine), as it is depicted in Figure 2(a).

OpenFlow is used in order to mangle the traditional ARP process and establish the appropriate load balancing schemes. The authors provide their solution as an Information Centric approach, backwards compatible and directly applicable to traditional IP-based networks, where the IP address is not used as a unique identifier of each host but is characterizing a specific content. In this sense, a single host like for example a video streaming server, would feature several IP-addresses, one for each content that it provides. Since the interconnected networks are IP-compatible, the traditional procedure when searching for a host is performed; the requesting client issues an ARP request message and awaits for the ARP reply with the destination MAC address. Subsequently, an ARP request will trigger multiple ARP replies in the same broadcast domain. The intermediate OpenFlow equipment, with the respective OpenFlow controller, intercepts these replies and according to the load balancing scheme imposed, forwards the respective ARP reply. Each load balancing scheme that maps appropriately a set of requesting clients to a group of identical servers, sharing among them the aggregate traffic from these clients, deals with a variety of architectural challenges. The investigated architecture assumes that the content identifier is a URL, as the Content Delivery Networks do, and leave the end-points to be untouched operating as usually. The modifications apply on the underlying OpenFlow network, which intercepts the ARP messaging process in order to implement a variety of load balancing policies.

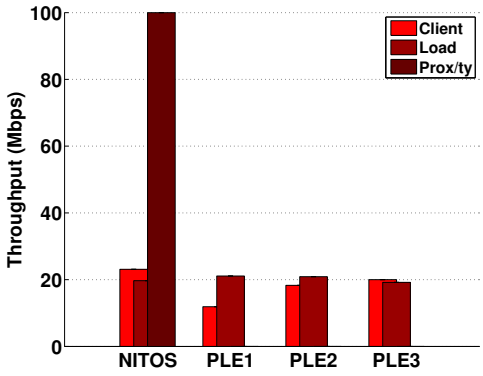
Thus, the adopted topology is involving two wireless virtual OpenFlow switches, with multiple wireless clients requesting a content from servers deployed in several sites in Europe. Physical wired OpenFlow switches are also employed at the different sites. The experimental topology and the content requesting applications are automatically deployed by using our aforementioned extensions in OMF6. The OpenFlow switches (virtual and hardware) are instructed to apply three different load-balancing policies. One policy, named Client-based policy, forwards each client to a specific server, that is chosen when the client first joins the network (Figure 2(c)). The Load-based policy periodically redirects some clients trying to unload the most congested servers (Figure 2(d)), while the Proximity-based policy just forwards the ARP reply from the most approximate server to the client (Figure 2(e)). In Figure 2(b) we present our findings for these three policies. As we can see, the Proximity-based policy allocates all the requests to one server, and thus the maximum throughput is received at one end point (NITOS server). On the other hand, the Load-based and Client-based policies manage to distribute the requests among the different servers (NITOS, PLE1, PLE2, PLE3).

B. Enhancing the wireless access network with cooperative diversity provided by mesh networks

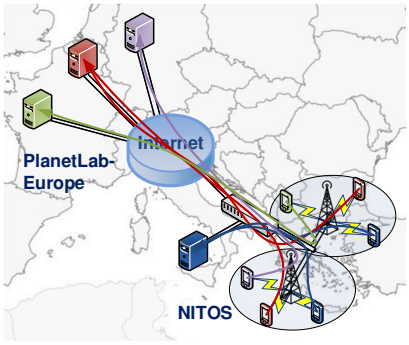
The focus of this scenario is the enhancement of the access network architecture. Normally, using the standardized protocol stacks that are employed in commercial devices, wireless networks with infrastructure (access-points) do not allow the cooperation among the stations. On the other hand, a wireless mesh architecture implies that there are no access-points and stations but peers and gateways, enabling the cooperation



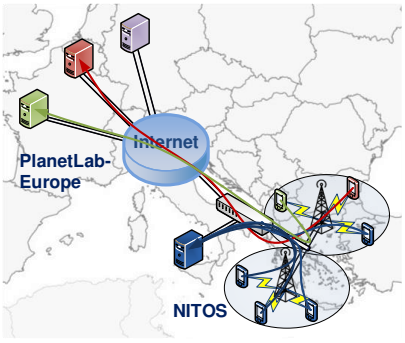
(a) The experiment topology.



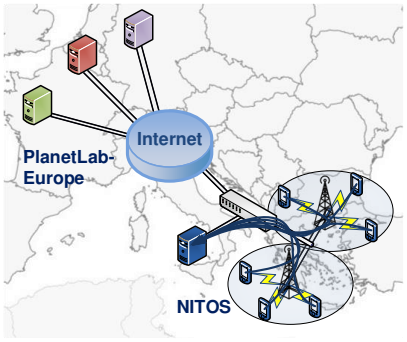
(b) Throughput performance of the three load balancing policies.



(c) Server Request mapping for the Client-based policy



(d) Server Request mapping for the Load-based policy



(e) Server Request mapping for the Proximity-based policy

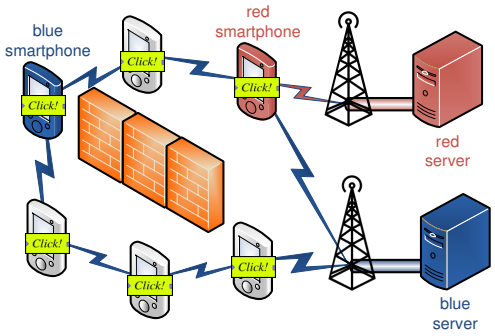
Fig. 2. European-wide private network (VPN) based on Open-vSwitch. Each color is mapping a server-client pair.

among the peers. The wireless mesh is an emerging architecture that is standardized (802.11s, Roofnet) and enables better performance in wireless networks when the users are crowded or mobile. The efficiency of a wireless mesh depends on the utilized routing protocol.

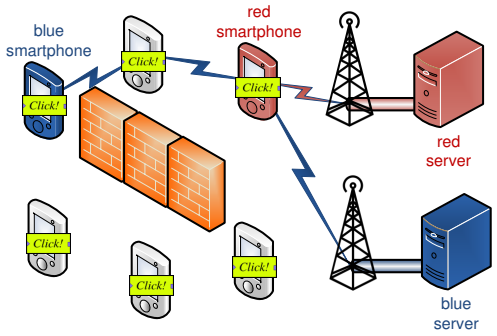
Shortest path routing is a conceptually simple and widely used approach, since there are several routing protocols that implement it. These protocols aim at achieving minimum end-to-end delay and maximum throughput performance, assuming that the network is not congested. However, the case of network congestion is a frequent phenomenon which should be seriously considered and appropriately handled. This is the motivation behind multi-path routing research, where alternative paths to the destination are utilized in order to provide resistant and seamless communication. Theory driven research on this area has concluded in a scheme, named Back-Pressure [16], which achieves throughput optimality under some prerequisites. In particular, Back-Pressure operates on a time-slotted and centralized schedule, introducing a specific scheduling and routing policy. Additionally, there is an improved version, named Enhanced Back-Pressure, that can be configured to move packets in the direction of their shortest paths, improving the delay performance. However, both schemes have not been implemented and used in realistic networks, mainly because of the impracticable centralized scheduling policy and the time-slotted assumption.

Multiple research efforts focus on designing wireless and distributed routing algorithms, that implement as many as possible of the Enhanced Back-Pressure aspects and are applied in real-life wireless mesh networks. One of these works is presented in [15], where the authors propose a scheme that incorporates most of the Enhanced Back-Pressure (EBoW) aspects in a WiFi compliant fashion. In particular, they introduce a scheme in which every node attempts to forward packets to less loaded and closer to the destination neighbors. The proposed scheme is using multiple paths to forward the wireless traffic, thus superseding other state-of-the-art multi-path routing algorithms (Horizon and CDP [15]), as well as the well known shortest-path SRCR algorithm [14]. The experimentation on these protocols performance is based on the OMF6 framework and its enhancements for Click Modular Router support. OMF6 controls the nodes and enforces them to behave as Back-Pressure routers, using the Click software tool. Figure 3(a) illustrates the experimentation topology shaped on the NITOS platform with use of OMF6.

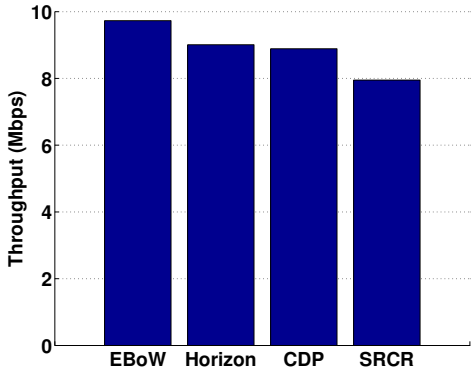
The integration of Click with OMF6 enables the configuration of this scheme in a large scale topology, similar to the access network of a realistic topology such as this of Figure



(a) An enhanced wireless access network with cooperative diversity. EBoW [15] uses efficiently both paths for the forwarding of the traffic generated by the blue smartphone and forwarded to the blue server.



(b) SRCR [14] uses only the shortest path to forward the traffic generated by the blue smartphone, which passes through the overloaded red smartphone.



(c) Throughput performance of the four under investigation routing algorithms.

Fig. 3. Wireless mesh network based on Click Modular Router. The wireless links indicate which paths are used for the traffic forwarding.

2(a), which the authors employed to evaluate their scheme. The evaluation of the proposed EBoW scheme, as well as its comparison with the other state-of-the-art wireless routing algorithms, is presented in Figure 3(c). The clear advantage of the EBoW scheme can be seen in terms of throughput network performance.

V. Conclusion

In this article, we describe the implementation of our SDN-specific extensions in the state-of-the-art control and management framework for wireless testbeds. We discuss the necessity and feasibility for experimentation using joint wireless and wired SDN, and the problems in testbed administration that we overcome using our contributions. Through the development and evaluation of realistic large scale scenarios with the use of the enhanced framework, we show the significant benefits of our developments. Our ongoing work includes further SDN extensions for the OMF6 framework, as well as the evaluation of more use cases and scenarios, in a large scale federated environment.

Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant agreement $N^{\circ}611165$, named SmartFIRE.

References

[1] OpenLab: The OpenLab federation of various European testbeds, http://www.ict-openlab.eu.
[2] FIBRE: Future Internet testbeds experimentation between Brazil and Europe, http://www.fibre-ict.eu.
[3] Fed4FIRE: Federation of European facilities for FIRE, http://www.fed4fire.eu.
[4] SmartFIRE: Enabling SDN Experimentation in Wireless Testbeds exploiting Future Internet Infrastructures in South Korea and Europe, http://eukorea-fire.eu.
[5] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM*, 2008.
[6] Thierry Rakotoarivelo, Maximilian Ott, Guillaume Jourjon, and Ivan Seskar. OMF: a control and management framework for networking testbeds. *ACM SIGOPS*, 2010.
[7] GENI: Exploring Network of the Future, http://www.geni.net.
[8] Expedient: A Pluggable Platform for GENI Control Frameworks, http://yuba.stanford.edu/~jnaous/expedient.

[9] Donatos Stavropoulos, Aris Dadoukis, Thierry Rakotoarivelo, Maximilian Ott, Thanasis Korakis and Leandros Tassiulas. Design, Architecture and Implementation of a Resource Discovery, Reservation and Provisioning Framework for Testbeds. *WiNMeE*, 2015.

[10] Kostas Choumas, Nikos Makris, Thanasis Korakis, Leandros Tassiulas and Maximilian Ott. Exploiting Openflow Resources towards a CCLAN. *IEEE EWSDN*, 2013.

[11] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. Can the production network be the testbed? *USENIX OSDI*, 2010.

[12] OvS: Open virtual Switch, http://openvswitch.org.

[13] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 2000.

[14] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. *ACM MobiCom*, 2005.

[15] Kostas Choumas, Thanasis Korakis, Iordanis Koutsopoulos, and Leandros Tassiulas. Implementation and End-to-end Throughput Evaluation of an IEEE 802.11 Compliant Version of the Enhanced-Backpressure Algorithm. *EAI TRIDENTCOM*, 2012.

[16] Michael J. Neely, Eytan Modiano, and C.E. Rohrs. Dynamic Power Allocation and Routing for Time Varying Wireless Networks. *IEEE INFOCOM*, 2003.