An OpenFlow Architecture for Energy Aware Traffic Engineering in Mobile Networks

Carlos Donato, Pablo Serrano, Antonio de la Oliva, Albert Banchs, Carlos J. Bernardos

***Abstract*—To cope with the growing traffic demand, future mobile networks will be denser and integrate heterogeneous technologies. However, if not properly engineered, such networks may incur in huge energy wastage when there is little traffic, and may suffer from an unbearable management burden caused by the variety of technologies integrated. In this paper, we propose and implement a novel management architecture for mobile networks based on OpenFlow, which supports resource-on-demand provisioning in a centralised control plane, and hides the technology specifics to the controller through the use of abstractions. The feasibility of the approach is demonstrated by a real-life prototype based on commercial, off-the-shelf devices.**

***Index Terms*—OpenFlow, SDN, WLAN**

I. Introduction

Busy-hour Internet traffic is expected to grow more rapidly than average Internet traffic [1], which requires that future networks support a fast growing data volume that fluctuates over time and space, because of users behaviour and mobility. The solution to cope with this growing traffic demand necessarily entails using more "points of attachment" (PoAs) such as 802.11 Access Points (APs) or LTE Evolved Nodes B (eNBs), by increasing their density and by using different wireless technologies, as well as offloading the network infrastructure (through e.g., *device-to-device communication*) [2]. However, such a deployment substantially complicates the management and traffic engineering (TE) of the network. Furthermore, having a large number of deployed PoAs also influences the energy cost; indeed, today's APs and base stations running at zero-load consume almost as much energy as when running at full capacity [3].

Over the last few years, the Software Defined Networking (SDN) paradigm has gained a lot of popularity. SDN decouples the system that makes decisions about where traffic is sent (i.e., control plane) from the underlying system that forwards traffic to the selected destination (i.e., data plane). With SDN, network administrators can program the behaviour of the network in a centralised way, without requiring to access and configure each of the hardware devices independently, which greatly simplifies overall network management.

Among the different protocols that can be used to realise the SDN concept, OpenFlow [4] is the one most widely adopted by the telecommunications industry. While so far OpenFlow efforts have been mostly focused on wired networks, and comparatively little attention has been paid to applying it to wireless, the potential benefits of this technology extended to adequately support mobile networks are much higher than with wired networks, for the following reasons:

- It allows to manage all the heterogeneous technologies (such as cellular or WLAN) in a transparent way, hiding the specificities of the technologies to the TE tool.
- It allows to integrate in a single decision point different types of communication that have traditionally been handled separately, such as device-to-device and infrastructure communications.
- It can also be used to address new functions that are not needed in wired networks, such as switching off those points of access that are not needed at a given point in time, and thus reduce energy consumption.
- Finally, using an adequate abstraction of wireless resources, handling the complex capacity calculations in heterogeneous deployments could be as simple as in wired networks.

In this paper we propose and implement the OFTEN framework (Open Flow framework for Traffic Engineering in mobile Networks with energy awareness), which addresses all the issues mentioned above. We envision a centralised controller that (a)periodically performs TE optimisations based on a given set of policies. By using a common API for all technologies, this controller issues commands that are oblivious to the technologies underneath. Furthermore, the controller does not only serve to manage the mobile network infrastructure but can also reach mobile terminals and even trigger device-to-device communications between them. The goals of the paper are aligned with some of the efforts currently being done at the Wireless and Mobile Working Group of the ONF,[1] where extensions to OpenFlow for mobile network architectures are being studied. We are actually pushing some of these extensions through the EU project iJOIN.[2]

To the best of our knowledge, this is the first attempt to implement a working prototype of OpenFlow that provides all these features and can be used in real networks. Indeed, while there have been several approaches proposed to manage heterogeneous mobile networks, they all suffer from one or more of the following limitations: (i) they propose the general framework but do not fully specify and implement the underlying architecture [5]; (ii) they centralise the configuration but work at a much coarser time-scales [6]; and (iii) they do not seamlessly integrate different wireless technologies nor manage end-terminals in an integrated manner [7].

All authors are with Univ. Carlos III de Madrid, Spain. C. Donato and A. Banchs are also with Institute IMDEA Networks, Spain.

This work has been partly supported by the European Community through the iJOIN (FP7-ICT-317941) and the CROWD (FP7-ICT-318115) projects. Apart from this, the European Commision has no responsibility for the content of this article.

[1]Open Networking Foundation: http://opennetworking.org/

[2]http://www.ict-ijoin.eu/

II. Architecture

We propose the architecture illustrated in Fig. 1, which consists of a number of technology-agnostic elements plus some technology-specific modules. We start with the former:

- The first element is the database containing the current **network vision**, i.e., the *nodes* of the network, the active *links* connecting them, the potential links that can be used and their capacity as well as the traffic demand.
- Based on this network vision, the **optimiser** module runs (a)periodically (e.g., every 5 minutes or after a major change in network conditions) to obtain the configuration that maximises performance, according to a set of given policies that trade-off energy consumption and performance.
- The configuration resulting from the optimiser module is passed to the **controller** via the *northbound* interface; this interface is not shown in Fig. 1 and depends on the specific OpenFlow controller chosen (we leave it outside the scope of our architecture).
- Finally, the controller implements this configuration through **OpenFlow++**, a technology-agnostic interface that extends the OpenFlow protocol (OFPT) to support the required functionality for mobile networks (described in the following sections).

One key feature of the proposed architecture is the ability of the upper layer modules described above to operate on a technology-agnostic manner, which allows to (i) manage the entire heterogenous network in an integrated way, allowing for a joint optimisation of all the technologies; and (ii) easily adapt network operation to new requirements/objectives, by simply modifying the technology-agnostic modules. This functionality is enabled by the following lower layer modules, which are technology-specific:

- A **technology dispatcher**, which is responsible to re-direct the OpenFlow++ primitives to appropriate technology-specific module.
- Technology-specific **mapping modules**, which convert the OpenFlow primitives into the primitives of the corresponding technology and viceversa; among others, this includes the setting of new configurations and update the network vision.

Another key feature is that the architecture can be extended to support new technologies in a non-disruptive way: to enable the use of a new technology, we only need to design the specific mapping module and a minor extension to the Dispatcher to identify when a command is from/to that specific technology. We note that, because of this centralised management of all technologies in the network, the controller might suffer from scalability issues, a general concern in SDN scenarios. To ease this burden, one possible solution would be the definition of hierarchical areas [8], [9].

We next describe the technology-agnostic operation of the upper modules and how the OpenFlow++ interface is used and extended to support this vision (we summarise in Table I the required extensions to OpenFlow), while the technology-specific operation is described in the next section.

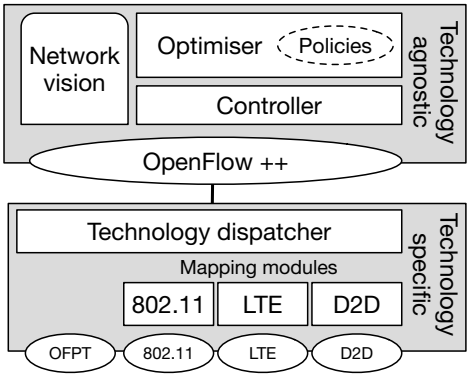

Fig. 1. Open Flow-based SDN architecture for heterogeneous networks.

Primitive/parameter	Use by OFTEN
`OFPT_HELLO`	Announce new nodes and links. Link unavailability is signalled with `TCP FIN`.
`OFPT_FEATURES_REPLY`	Announcement of non-OF features (e.g. de-activation) via the `reserved` field
`OFPT_FLOW_MOD`	Issue mobility commands, i.e., change point of attachment
`OFPT_EXPERIMENTER`	Switching on and off of resources
`OFPT_METER_MOD`	Add/remove meter configuration
`curr_speed`, `max_speed`	Capacity announcement
`counters`, `meter_bands`	Measure throughput and trigger optimization module

TABLE I
Extensions to OpenFlow introduced by OFTEN.

A. Technology-agnostic operation

One of the key challenges of our architecture is to achieve a technology-agnostic vision of the network. This allows the optimiser and controller to operate on abstract "nodes" and "links" instead of, e.g., 802.3az links or eNBs. More specifically, the challenge is to map the actual physical network, like e.g. the one illustrated in Fig. 2 (top), into an abstract vision (bottom). While the physical network is composed of wired links in the backhaul, relatively stable links from mobile terminals to cell towers, higher capacity but more dynamic connections to 802.11 Access Points, potential device-to-device links, etc., in the abstract vision there are nodes and links, each with a different capacity, that can be reconfigured and switched on and off as required by the optimiser.

The proposed architecture is flexible to accommodate different TE mechanisms for the optimiser, depending on the computational resources availability, complexity of the network, and periodicity/timeliness of the operation of the optimisation (see Section IV.C for a description of the mechanism used in our implementation). Furthermore, the optimisation of the network does not have to be changed whenever a new technology is introduced, and only requires mapping the features of the new technology to our abstractions.[3]

The mapping between the actual network and the abstract one is made in the technology dispatcher module, which

[3]Of course, to achieve this it is critical that our abstraction is general enough to cover the functionality provided by the new technology, as otherwise the optimiser would need to be adapted to the novel functionality provided.

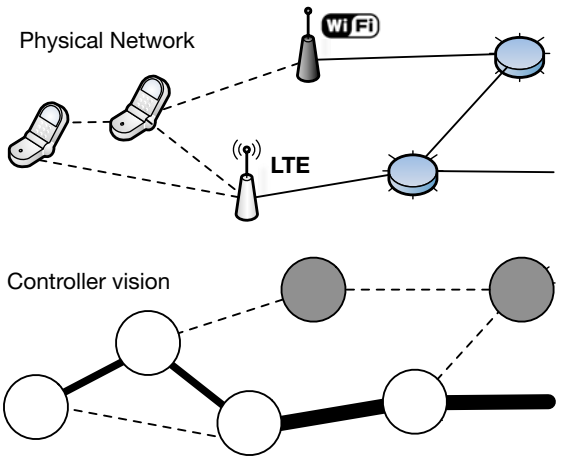

Fig. 2. Physical network (top) and abstract vision (bottom). Grey circles denote nodes in power-saving mode, dotted lines represent available links, and solid lines represent used links.

performs the association between the mapping modules and the OpenFlow++ interface. This interface is the central element of the architecture that supports the following functionality with technology-independent primitives: (i) the maintenance of the information in the database, including changes to link capacities, reachability, etc., (ii) the control of the forwarding tables, and (iii) the (de)activation of resources. We next describe the OpenFlow commands upon which we rely, as well as the extensions required, to support these features.

B. Maintaining the network vision

The first challenge is to maintain the list of nodes that compose the network, which in our case appear and disappear more frequently than in "traditional" (wired) networks, due to users' mobility and wireless propagation. For the case of nodes connected to the network, i.e., APs or eNBs, they just have to establish a connection to the default OpenFlow transport protocol `6653` via TLS or TCP, and then perform the usual `OFPT_HELLO` exchange. For those nodes that can be (de)activated, we need to extend the database to announce this feature, which we do via the `reserved` field in the `OFPT_FEATURES_REPLY` message.

For the case of nodes that are one or more hops away from the wired infrastructure, we do not require them to implement any (major) modification and, in particular, to support OpenFlow. However, as they have to appear as nodes in our vision, we require that the PoAs act on the behalf of the terminals, and register them with the controller (following the standard procedure) whenever they detect there is a new node or a new candidate link. When no link towards a terminal is available, the connection is terminated via e.g. a `TCP FIN` message. Thus, in order to keep the list of nodes available in the network, our architecture does not need to introduce changes to the OpenFlow specification, but only to ensure that the database of (de)registered nodes is updated when required.

Similarly, in order to announce the capacity of the links that connect two or more nodes, we can rely on the structures already defined by the OpenFlow specification, i.e., the `curr_speed` and `max_speed` parameters, which define the current and maximum bitrates, respectively, of a port. Whenever the capacity of a given wireless link changes (because of e.g. WLAN interference, a node moves away from the eNB), the node responsible for a link needs to modify the value of the corresponding parameters.

Finally, the usage of the links carrying data can be easily tracked with per-flow `counters`. By periodically polling these counters with read operations, the database can identify which links are becoming congested and which ones are being under-occupied and could support more traffic. This can then trigger the corresponding optimisation. We note that an alternative implementation could be based on the use of per-flow `meters`; indeed, by specifying `meter bands`, we can trigger an action when certain thresholds are passed.

C. Installing a new configuration

For wired links, the setting of forwarding paths does not require to modify the default OpenFlow operation. With our abstraction, a wireless terminal can be considered as a node with a number of ports but only one active forwarding entry at a time, which corresponds to the selected point of attachment. In this way, changing the point of attachment only requires modifying a flow entry via the `OFPT_FLOW_MOD` message, e.g. to change of the Access Point the node is associated with, or the use of the cellular link. As we describe next, the Technology Dispatcher decides the technology specific module that handles the commands and triggers the protocol-specific operations to implement the change. As in the previous case, there is not need to specify new OpenFlow primitives.

D. Switching on/off resources

Energy-efficient operation of a network requires the ability to power on and off resources as required [10]. Accordingly, our architecture has been designed to provide such support. For the corresponding set of operations, we cannot rely on or extend the default OpenFlow primitives, and therefore we need to specify new primitives. To do this, we rely on the "experimenter" symmetric messages (`OFPT_EXPERIMENTER`), which are used for those nodes that upon registration announced that they supported deactivation, in addition to the time it takes to switch between states (so the the controller can preemptively activate resources) and the corresponding power consumption (to duly optimase energy consumption).

Following the above, we extend the OpenFlow set of primitives with a pair of commands, `switch_on` and `switch_off`, to power on and off (respectively) a given node. As we will describe in our testbed, these primitives can be used even when nodes do not support a sleep state, but are connected to e.g. a switched rack power distribution units (PDU), which can be remotely controlled via a network connection.

III. Mapping to technologies

The architecture enables an integrated traffic management of a heterogeneous mobile network, through the use of the OpenFlow++ primitives. We next describe how these primitives are

mapped back-and-forth into technology-specific functionality thanks to the technology dispatcher, which acts as a relay of the messages between the OpenFlow++ API and the modules doing the mapping. *In nuce*, we require the ability to:

- Detect when new nodes and links are available, including the potential points of attachment to the network.
- Estimate the available capacity of the wireless links.
- Change the point of attachment of a terminal without disrupting the traffic served.

In what follows, we describe how the above is supported by the dispatcher and the current wireless technologies, by just introducing minor extensions to their operation.

A. Technology dispatcher

New nodes or available links are announced in a technology-specific manner to this module (as detailed next), which then performs the translation to the OpenFlow++ API. With this module, a mobile terminal detected by a set of access points and an eNB (i.e., multiple announcements) is registered only once as a node, but with a set of candidate links towards existing nodes. Given that wired nodes may support different deactivation techniques (e.g., wake-on-LAN, switched power distribution units), this module needs to be aware of the specific technique supported in order to issue the corresponding commands when needed. The capacity of the wireless links is computed by each technology as described next, and then passed via the OpenFlow++ API using the parameters described above.

Changing the default forwarding table of a terminal corresponds to performing a handover, which can be intra- or inter-technology. The former is handled within each technology using its own mechanisms, while for the latter we rely on the 3GPP support for multiple radio access networks (RANs), including those that are non-cellular. The mobile 3GPP architecture centralises the support of inter-RAN handovers on the cellular network (more specifically, on a set of network entities that may act as control and data plane anchors for the different handover scenarios). This is based on the assumption of full cellular coverage.

B. 802.11 mapping

In Wi-Fi networks there are at least three mechanisms to detect when a link towards a mobile terminal is available: (i) the `probe request` messages the mobile periodically sends on different channels to detect known or new Access Points (APs), which can also trigger the presence of a new node that is duly reported to the technology dispatcher; (ii) passive scanning mechanisms; and (iii) the `Neighbour Report` message exchange from the recent 802.11k amendment, already supported by new devices such as e.g. iPhones, which is used by mobile terminals to learn about APs in the surroundings, and by APs to gather measurements about the quality of the channel towards other APs, which are then reported to the network.

The estimation of the capacity of a link is supported by these measurements, building on e.g. the usual mappings of signal quality to maximum throughput, which are reported to the network vision database. In case a group of nodes contend in a WLAN, the AP can derive the achievable capacity thanks to the use of the *linearised capacity* model proposed in [11], which abstracts the specifics (i.e., contention) of the channel access into a simple linear-based model.[4]

Changing the point of attachment of a Wi-Fi node while providing a seamless experience results is very challenging, given that in 802.11 (i) the mobile terminal typically selects its "best" point of attachment, and (ii) the signalling for carrier-grade operation is relatively complex. Still, thanks to the recent 802.11v and 802.11r amendments, this can be achieved with minor disruption of the service (this is validated by our prototype of Section IV). Fig. 3 illustrates a simplified version of the signalling occurring on the access network. Next, we describe how these interactions support the changing of the point of attachment when triggered by the OpenFlow command.

Following Fig. 3, when a terminal powers on, it authenticates and associates with the best AP (i.e., AP1), and performs the `Neighbour Report` exchange to learn about the APs in the vicinity belonging to the same network. During these operations, the APs that overhear the node activity report on the different links available to the mobile terminal, and update the database accordingly (among these, AP2 in the Figure). Assuming at some point that the optimiser decides that it is better if the mobile node (*MN*) associates with AP2, the controller issues a `OFPT_FLOW_MOD` primitive to change the (only) default forwarding entry of *MN*, from node *AP1* to node *AP2*. This primitive is processed by the 802.11 module with the controller, which issues a command to AP1 to trigger the 802.11v `BSS Transition Request` message, so the MN re-associates with AP2. Thanks to the use of 802.11r, this re-association is noticeably shorter than the original one. As we will see in the next section, the controller can duly issue other OpenFlow primitives to minimise the impact on performance.

C. Cellular mapping

We next describe the guidelines to implement a technology-specific module similar to the one described above for the case of cellular networks. While 3GPP-based networks involve more complex procedures, they also tend to favor a centralised control and estimation of resources, and therefore we expect that the design of this module is simpler than in the Wi-Fi case. However, given that cellular networks are designed for large deployments, one key difference with Wi-Fi is on maintaining user location.

If we focus on packet switched communications in a cellular network the list of active users within a cell is available at the Mobility Management Entity (MME), while the inactive users are less accurately located.

The above challenges the implementation of infrastructure-on-demand schemes, as a group of inactive users could suddenly change to be active in a certain geographical area, and an aggressive energy-saving policy might have powered off

[4]Note that the proposed linearised capacity model can also be used to abstract the capacity of any technology, including OFDM, CDMA, etc.

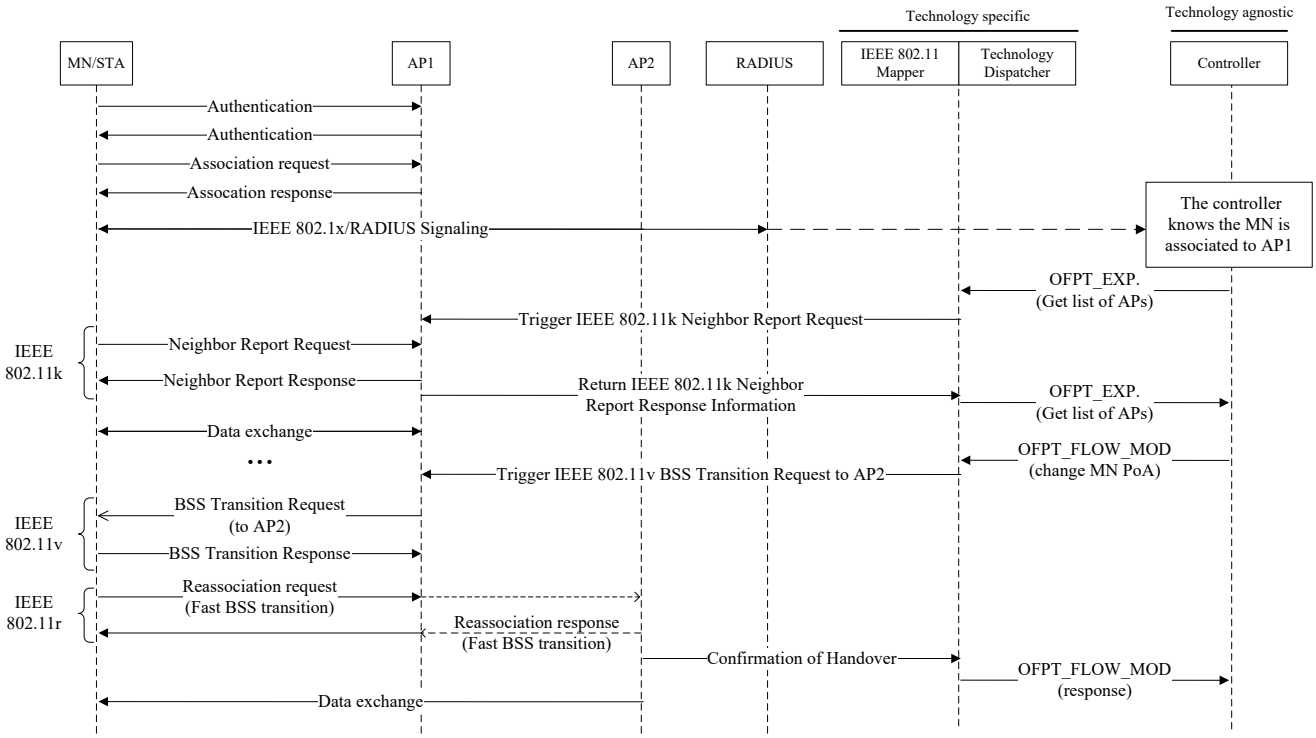

Fig. 3. Simplified handover operation.

too many Base Stations to promptly react to the demand. Except for this, cellular networks readily provide the means to support the requirements of the extended OpenFlow interface: active UEs periodically report the quality of the channel towards other Base Stations, and network-initiated handovers are supported.

D. Device to device mapping

Our architecture also supports device-to-device communications, where mobile nodes opportunistically share wireless links for a better performance. However, in contrast to the previous cases, this paradigm requires introducing modifications to the mobile terminals. To that aim, we have adapted our SOLOR framework [12], which builds on Wi-Fi Direct to opportunistically create D2D groups, to communicate with the technology dispatcher to announce link availability and support the set-up and release of this links. Given that SOLOR is a decentralised mechanism, the adaptation consists basically on centralising the control of the modules.

IV. Proof of Concept

To validate the feasibility of our approach, we have prototyped our architecture in a small testbed that includes the technology specific modules for Wi-Fi. The testbed, deployed in an office setting, is depicted in Fig. 4 and consists of one controller, an OpenFlow switch, two APs, two MNs, a correspondent node (CN) to support traffic generation, and a switched PDU to support the (de)activation of APs via HTTP requests.

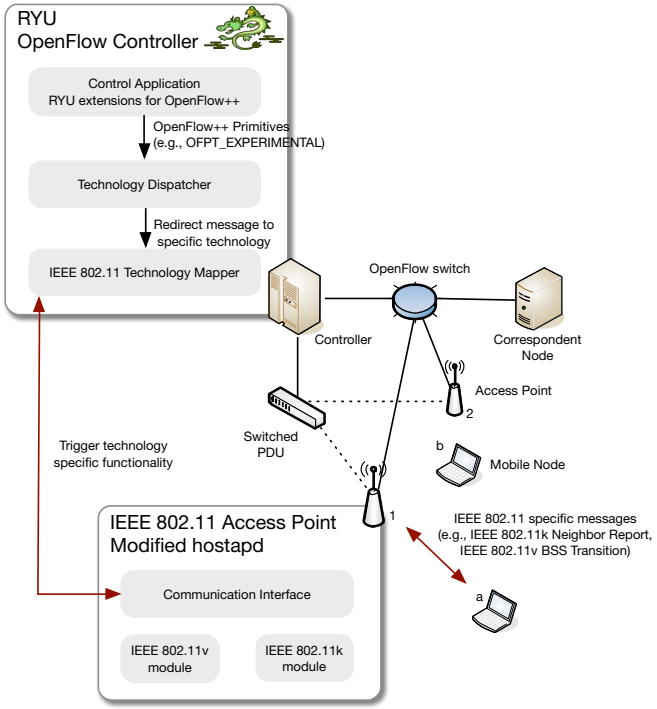

Fig. 4. Deployed testbed and implemented modules and interfaces.

A. Implementing seamless NIHO

One of the benefits of an OpenFlow-based architecture is that it supports the implementation of a variety of mobility protocols. For simplicity, we decided to implement the following sequence of commands whenever the controller decides to change the point of attachment of a MN: (1) activate at the switch *bicasting* of traffic between the CN port and the ports the APs are connected to; (2) issue the 802.11v BSS Transition Request; (3) wait until the handover is completed, and inform the controller accordingly; and (4) stop the bicasting of traffic at the switch and re-configure the forwarding tables accordingly.

B. Software setup

The controller is a desktop machine running Linux and the `Ryu` SDN framework,[5] which we extend to support the proposed `OpenFlow++` API, and two Python libraries: one to map the `switch_on/off` commands to the switched PDU, and another one for the 802.11-specific mechanisms. The controller also runs a `MySQL` database[6] to keep the network status. Our controller uses a variation of the TE algorithm that we designed in [11], which is based on an integer programming formulation and minimises the number of nodes required to support a set of flows at a reduced computational cost. Other TE schemes could be supported, based on, e.g., flow management policies or traffic measurement requirements (see [13] for a recent survey).

The APs are small PCs running Linux and extended to support the required functionality as follows. We have modified the widely used `hostapd` demon[7] to set up a connection with the 802.11 module at the controller, so that the AP can report changes in the network conditions (which are updated to the database) and can set up new configurations. When the controller issues a change on the default forwarding path of a node, the 802.11 module translates this primitive to an 802.11k request to perform channel measurements (so that the mobile updates the list of available APs) and an 802.11v command to change the point of attachment. An overview of the developed software modules and interfaces is depicted in Fig. 4.

The CN is a regular desktop machine, while the MNs are small PCs, like the switch, which runs `OpenVSwitch`. Finally, we set up an additional desktop machine, not shown in the picture, running the `FreeRADIUS` server[8] to enable the use of WPA2-enterprise as required by Wi-Fi Passpoint.

C. Supporting infrastructure on demand

To validate that the controller activates resources as traffic requirements vary, we perform the following experiment. We first generate a TCP flow between the CN and node a, which is associated with AP 1, while AP 2 is inactive. At $t = 20$ s, we generate another TCP flow between the CN and node b, which saturates the capacity of the link. Thanks to periodic measurements, the controller decides 3 s later that more resources are required, and switches on AP 2.[9] Once this AP is available, it notifies the controller, which then immediately issues a handover trigger to node b that associates with the new AP at $t = 45$ s.

The results of this experiment are depicted in Fig. 5, where we plot the per-flow throughput (bottom) and the total throughput (top) as measured by the Wireshark tool. According to the figure, the first TCP flow achieves at first about 20 Mbps, but when the second flow appears, its throughput its reduced to 5 Mbps, while the former gets about 15 Mbps (with some variations due to contention). Once this has been moved to AP 2, the flow from node a gets again about 20 Mbps, while the flow from node b gets about 20 Mbps as well.

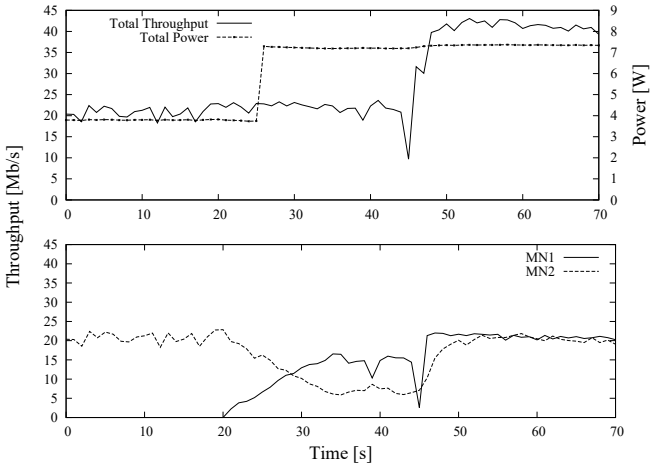

Fig. 5. Validation of the proof of concept.

We also represent in Fig. 5 the estimated energy consumption of the infrastructure, following the model of [14]. As compared with the case of both APs on, our solution reduces energy consumption by 18%, an improvement that comes at the cost of an increased delay, mainly because the time it takes to detect a new flow and power-up the AP. Following the results reported in [10], we estimate that our framework could reduce energy consumption by 30–40% in a campus-wide deployment (note that our experiment corresponds to an "upper bound" in terms of performance reduction, as there is always wireless activity).

V. Conclusions and Future Work

In this article, we have proposed a novel SDN architecture to support traffic engineering in mobile networks. Our OpenFlow-based architecture facilitates the support for heterogeneous technologies by making use of abstractions, and enables the use of infrastructure-on-demand schemes via novel primitives to switch on/off devices as required. The validity of our approach has been demonstrated for the case of 802.11 through a simple prototype.

[5] http://osrg.github.io/ryu/
[6] http://www.mysql.com
[7] http://w1.fi/hostapd/
[8] http://freeradius.org

[9] We note that the focus of our validation is on the framework and not on the specific mechanism to provide infrastructure on demand, which we acknowledge could be improved.

In addition to our standardisation activities discussed in Section I and the scalability analysis of the architecture described in Section II, we are working on the following challenges: (i) we are currently extending the prototype to LTE-like environments, by adapting the open source LTE/EPC Network Simulator LENA[10] to run on top of a Software Defined Radio, and (ii) we are also deploying our framework in a campus-size testbed, to evaluate the performance improvements that can be achieved,[11] analysing the trade-off between energy consumption and performance when using these resource-on-demand schemes [15].

REFERENCES

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2013-2018," Cisco, Tech. Rep., june 2014. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[2] W. Sun, O. Lee, Y. Shin, S. Kim, C. Yang, H. Kim, and S. Choi, "Wi-Fi Could Be Much More," *Communications Magazine, IEEE*, vol. 52, no. 11, pp. 22–28, November 2014.

[3] P. Serrano, A. de la Oliva, P. Patras, V. Mancuso, and A. Banchs, "Greening wireless communications: Status and future directions," *Computer Communications*, vol. 35, no. 14, pp. 1651 – 1661, 2012.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[5] C. Bernardos, A. de la Oliva, P. Serrano, A. Banchs, L. Contreras, H. Jin, and J. Zuniga, "An architecture for software defined wireless networking," *Wireless Communications, IEEE*, vol. 21, no. 3, pp. 52–61, June 2014.

[6] Y. Yiakoumis, M. Bansal, A. Covington, J. van Reijendam, S. Katti, and N. McKeown, "BeHop: A Testbed for Dense WiFi Networks," in *Proceedings of the 9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, ser. WiNTECH '14. New York, NY, USA: ACM, 2014, pp. 1–8.

[7] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards Programmable Enterprise WLANS with Odin," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 115–120.

[8] X. Li, P. Djukic, and H. Zhang, "Zoning for hierarchical network optimization in software defined networks," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–8.

[9] X. Li and H. Zhang, "Creating logical zones for hierarchical traffic engineering optimization in sdn-empowered 5g," in *Proceedings of the International Conference on Computing, Networking and Communication (ICNC), 2015*, 2015.

[10] F. Ganji, L. Budzisz, F. Debele, N. Li, M. Meo, M. Ricca, Y. Zhang, and A. Wolisz, "Greening campus WLANs: energy-relevant usage and mobility patterns," *Computer Networks, Special Issue on Green Communications*, 2014, accepted for publication. [Online]. Available: http://www.tkn.tu-berlin.de/fileadmin/fg112/Papers/2014/Ganji14greening_campus_wlans.pdf

[11] A. De La Oliva, A. Banchs, and P. Serrano, "Throughput and energy-aware routing for 802.11 based mesh networks," *Comput. Commun.*, vol. 35, no. 12, pp. 1433–1446, Jul. 2012.

[12] A. Garcia-Saavedra, B. Rengarajan, P. Serrano, D. Camps-Mur, and X. Costa-Perez, "SOLOR: Self-Optimizing WLANs With Legacy-Compatible Opportunistic Relays," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[13] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, no. 0, pp. 1 – 30, 2014.

[14] P. Serrano, A. Garcia-Saavedra, G. Bianchi, A. Banchs, and A. Azcorra, "Per-frame Energy Consumption in 802.11 Devices and its Implication on Modeling and Design," *ACM/IEEE Transactions on Networking, accepted for publication*.

[10] http://networks.cttc.es/mobile-networks/software-tools/lena/

[11] We have made our source code available so other researchers and practitioners can perform similar measurements: https://oruga.it.uc3m.es/redmine/projects/often

[15] J. Ortin, P. Serrano, and C. Donato, "Modeling the Impact of Start-Up Times on the Performance of Resource-on-Demand Schemes in 802.11 WLANs," in *Proceedings of the Fourth IFIP Conference on Sustainable Internet and ICT for Sustainability (SustainIT 2015)*, April 2015. [Online]. Available: http://www.it.uc3m.es/pablo/papers/impact-start-up-times.pdf

AUTHORS' BIOGRAPHIES

Carlos Donato (carlos.donato@imdea.org) received his B.Sc. in telecommunication engineering from Universidad Carlos III de Madrid (UC3M) in 2014, and his M.Sc. in telematics engineering from the same university in 2015. He is currently pursuing his PhD while working at Institute IMDEA Networks. His research focuses on performance evaluation and centralised optimisation of wireless networks.

Pablo Serrano [M] (pablo@it.uc3m.es) received his M.Sc. and Ph.D. degrees in telecommunications from UC3M in 2002 and 2006, respectively, where he currently holds the position of Associate Professor. He has over 60 scientific papers in peer-reviewed international journal and conferences. He serves on the Editorial Board of IEEE Communications Letters, has been a guest editor for Computer Networks, and has served on the TPC of a number of conferences and workshops including IEEE INFOCOM, IEEE WoWMoM and IEEE Globecom.

Antonio de la Oliva [M] (aoliva@it.uc3m.es) received his telecommunications engineering degree in 2004 and his PhD. in 2008 from UC3M, where he has been working as associate professor since. His current line of research is related to networking in Extremely Dense Networks. He is an active contributor of IEEE 802 where he has served as Vice-chair of IEEE 802.21b and Technical Editor of IEEE 802.21d. He has also served as Guest Editor of IEEE Communications Magazine.

Albert Banchs [SM] (banchs@it.uc3m.es) received his degree in telecommunications engineering from the UPC BarcelonaTech in 1997, and his Ph.D. degree from the same university in 2002. He was a visiting researcher at ICSI, Berkeley, in 1997, worked for Telefonica I+D in 1998, and for NEC Europe from 1998 to 2003. He has been with UC3M since 2003. Since 2009, he also has a double affiliation as Deputy Director of IMDEA Networks.

Carlos J. Bernardos (cjbc@it.uc3m.es) received a telecommunication engineering degree in 2003 and a Ph.D. in telematics in 2006, both from UC3M, where he worked as a research and teaching assistant from 2003 to 2008 and, since then, as an associate professor. His current work focuses on virtualization in heterogeneous wireless networks. He has published over 70 scientific papers in international journals and conferences, and he is an active contributor to the IETF. He has served as Guest Editor of IEEE Network.