An OpenFlow Architecture for Energy-Aware Traffic Engineering in Mobile Networks

Carlos Donato, Pablo Serrano, Antonio de la Oliva, Albert Banchs, and 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 huge energy waste when there is little traffic, and may suffer from an unbearable management burden caused by the variety of technologies integrated. In this article, we propose and implement a novel management architecture for mobile networks based on OpenFlow, which supports resource-on-demand provisioning in a centralized control plane, and hides the technology specifics from 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.

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' behavior 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 using different wireless technologies, as well as offloading the network infrastructure (e.g., through *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 behavior of the network in a centralized way, without requiring that each device be accessed and configured independently, which greatly simplifies overall network management.

The authors are with Universidad Carlos III de Madrid.

Carlos Donato and Albert Banchs are also with IMDEA Networks Institute.

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

Among the different protocols that can be used to realize 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 all the heterogeneous technologies (e.g., cellular or WLAN) to be managed in a transparent way, hiding the specificities of the technologies to the traffice engineering (TE) tool.
- It allows different types of communication that have traditionally been handled separately to be integrated at a single decision point, 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 PoAs that are not needed at a given point in time, thus reducing 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 article we propose and implement the Open Flow Framework for Traffic Engineering in Mobile Networks with Energy Awareness (OFTEN), which addresses all the issues mentioned above. We envision a centralized controller that (a)periodically performs TE optimization based on a given set of policies. By using a common application programming interface (API) for all technologies, this controller issues commands that are oblivious to the technologies underneath. Furthermore, the controller not only serves to manage the mobile network infrastructure, but can also reach mobile terminals and even trigger device-to-device communications between

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

0890-8044/15/$25.00 © 2015 IEEE

them. The goals of the article 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:

- They propose a general framework but do not fully specify and implement the underlying architecture [5].
- They centralize the configuration but work on much coarser timescales [6].
- They do not seamlessly integrate different wireless technologies or manage end terminals in an integrated manner [7].

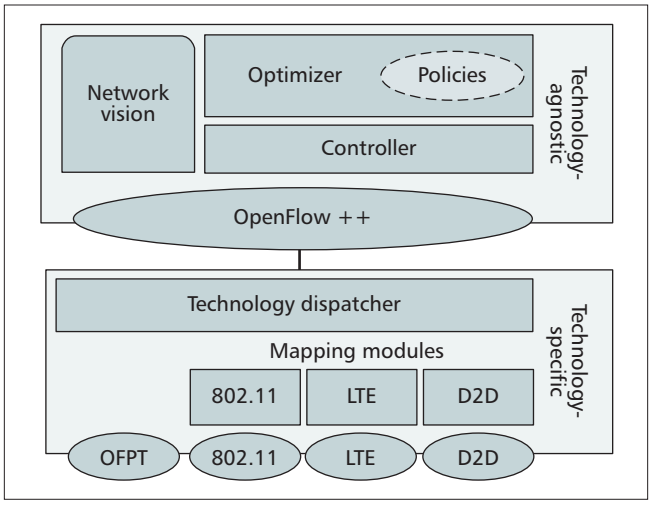

Figure 1. OpenFlow-based SDN architecture for heterogeneous networks.

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*, that is, 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 *optimizer* module runs (a)periodically (e.g., every 5 minutes or after a major change in network conditions) to obtain the configuration that maximizes performance, according to a set of given policies that trade off energy consumption and performance.
- The configuration resulting from the optimizer 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 in a technology-agnostic manner, which allows the entire heterogenous network to be managed in an integrated way, allowing joint optimization of all the technologies; and easily adaptation of 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 for redirecting the OpenFlow++ primitives to an appropriate technology-specific module
- Technology-specific *mapping modules*, which convert the OpenFlow primitives into the primitives of the corresponding technology and vice versa, including setting new configurations, updating the network vision, and more

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 centralized 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 summarize in Table 1 the required extensions to OpenFlow), while the technology-specific operation is described in the next section.

Technology-Agnostic Operation

One of the key challenges of our architecture is to achieve a technology-agnostic vision of the network. This allows the optimizer and controller to operate on abstract nodes and links instead of, for example, IEEE 802.3az links or eNBs. More specifically, the challenge is to map the actual physical network, such as 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 APs, potential device-to-device links, and so on, 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 optimizer.

The proposed architecture is flexible to accommodate different TE mechanisms for the optimizer, depending on the computational resources availability, complexity of the network, and periodicity/timeliness of the operation of the optimization (see the next section for a description of the mechanism used in our implementation). Furthermore, the optimization 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 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:

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

[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 optimizer would need to be adapted to the novel functionality provided.

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

Table 1. Extensions to OpenFlow introduced by OFTEN.

- The maintenance of the information in the database, including changes to link capacities, reachability, and so on
- The control of the forwarding tables
- The (de)activation of resources

We next describe the OpenFlow commands on which we rely, as well as the extensions required to support these features.

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 toward a terminal is available, the connection is terminated via, for example, 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, that is, the curr_speed and max_speed parameters, which define the current and maximum bit rates, respectively, of a port. Whenever the capacity of a given wireless link changes (e.g., a node moves away from the eNB due WLAN interference), the node responsible for a link needs to modify the value of the corresponding parameters.

Finally, the usage of the links carrying data can easily be 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 optimization. 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.

Installing a New Configuration

For wired links, the setting of forwarding paths does not require modification of 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, for example, to change of the Access Point with which the node is associated, 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 no need to specify new OpenFlow primitives.

Switching Resources On/Off

Energy-efficient operation of a network requires the ability to power resources on and off 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 controller can preemptively activate resources) and the corresponding power consumption (to duly optimize 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 describe in our testbed, these primitives can be used even when nodes do not support a sleep state, but are connected to, for example, a switched rack of power distribution units (PDUs), which can be remotely controlled via a network connection.

Mapping to Technologies

The architecture enables 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. Simply put, we require the ability to:

- Detect when new nodes and links are available, including the potential PoAs 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.

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 APs and an eNB (i.e., multiple announcements) is registered only once as a node, but with a set of candidate links toward 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 Third Generation Partnership Project (3GPP) support for multiple radio access networks (RANs), including those that are non-cellular. The mobile 3GPP architecture centralizes 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.

IEEE 802.11 Mapping

In WiFi networks there are at least three mechanisms to detect when a link toward a mobile terminal is available:

- The `probe request` messages sent to the mobile periodically, sending on different channels to detect known or new APs, which can also trigger the presence of a new node that is duly reported to the technology dispatcher
- Passive scanning mechanisms
- The `Neighbour Report` message exchange from the recent 802.11k amendment, already supported by new devices such as iPhones, which is used by mobile terminals to learn about APs in the area, and by APs to gather measurements about the quality of the channel toward other APs, which are then reported to the network

The estimation of the capacity of a link is supported by these measurements, building on, for example, 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 *linearized 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 PoA of a WiFi node while providing a seamless experience is very challenging, given that in 802.11 the mobile terminal typically selects its "best" PoA, and the signaling for carrier-grade operation is relatively complex. Still, thanks to the recent IEEE 802.11v and 802.11r amendments, this can be achieved with minor disruption of the service (this is validated by our prototype in the next section). Figure 3 illustrates a simplified version of the signaling occurring on the access network. Next, we describe how these interactions support the changing of the PoA 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 that at some point the optimizer decides it is better if the mobile node (MN) associates with AP2, the controller issues an `OFPT_FLOW_MOD` primitive to change the (only) default forwarding entry of the 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 IEEE 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 seen in the next section, the controller can duly issue other OpenFlow primitives to minimize the impact on performance.

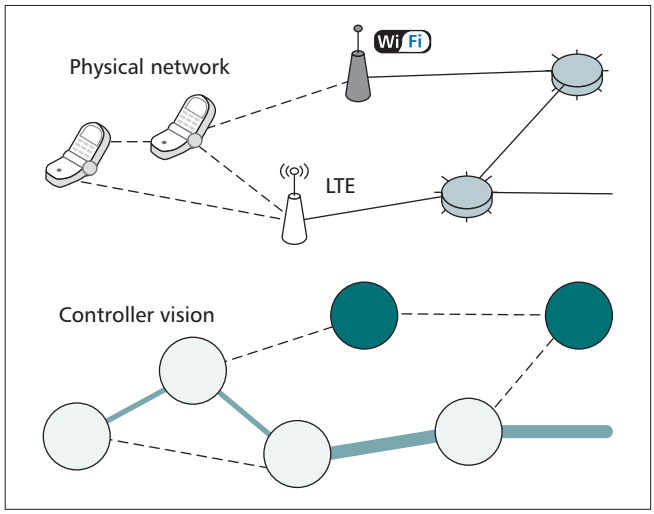

Figure 2. Physical network (top) and abstract vision (bottom). Gray circles denote nodes in power-saving mode, dotted lines represent available links, and solid lines represent used links.

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 centralized control and estimation of resources, and therefore we expect the design of this module to be simpler than in the WiFi case. However, given that cellular networks are designed for large deployments, one key difference with WiFi is in 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 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 being active in a certain geographical area, and an aggressive energy-saving policy might have powered off 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 toward other base stations, and network-initiated handovers are supported.

Device-to-Device Mapping

Our architecture also supports device-to-device communications, where mobile nodes opportunistically share wireless links for better performance. However, in contrast to the pre-

[4] Note that the proposed linearized capacity model can also be used to abstract the capacity of any technology, including orthogonal frequency-division multiplexing (OFDM), code-division multiple access (CDMA), and so on.

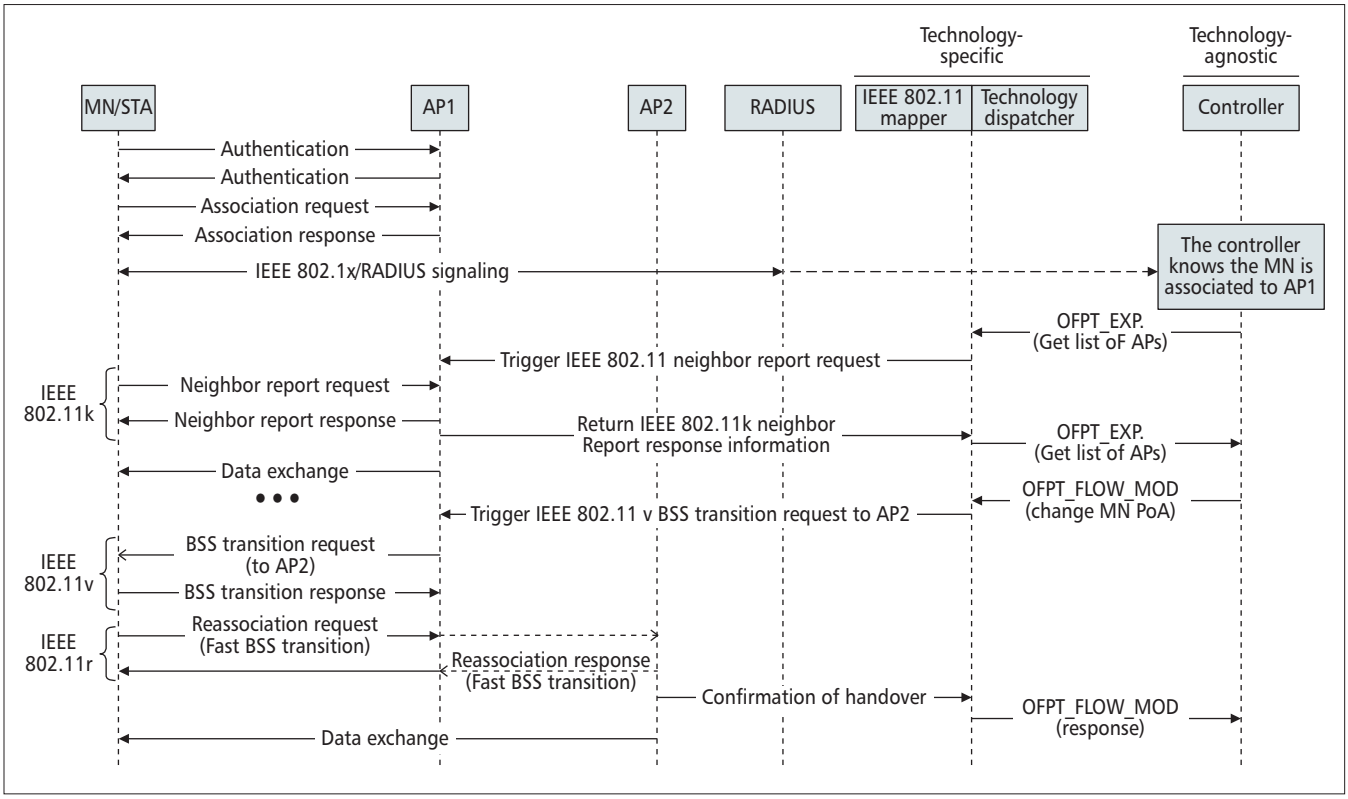

Figure 3. Simplified handover operation.

vious cases, this paradigm requires introducing modifications to the mobile terminals. To that aim, we have adapted our SOLOR framework [12], which builds on WiFi Direct to opportunistically create D2D groups to communicate with the technology dispatcher to announce link availability, and support the setup and release of this links. Given that SOLOR is a decentralized mechanism, the adaptation basically consists of centralizing the control of the modules.

Proof of Concept

To validate the feasibility of our approach, we have prototyped our architecture in a small testbed that includes technology-specific modules for WiFi. 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 packet data unit (PDU) to support the (de)activation of APs via HTTP requests.

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 PoA of an MN:

1. Activate at the switch bicasting of traffic between the CN port and the ports to which the APs are connected.
2. Issue the 802.11v base station system (BSS) transition request.
3. Wait until the handover is completed, and inform the controller accordingly.
4. Stop the bicasting of traffic at the switch and reconfigure the forwarding tables accordingly.

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 for the IEEE 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, for example, 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 set up new configurations. When the controller issues a change in 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 PoA. 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 WiFi Passpoint.

[5] http://osrg.github.io/ryu/

[6] http://www.mysql.com/

[7] http://w1.fi/hostapd/

[8] http://freeradius.org/

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 at first achieves about 20 Mb/s, but when the second flow appears, its throughput is reduced to 5 Mb/s, while the former gets about 15 Mb/s (with some variations due to contention). Once this has been moved to AP 2, the flow from node *a* again gets about 20 Mb/s, while the flow from node *b* gets about 20 Mb/s as well.

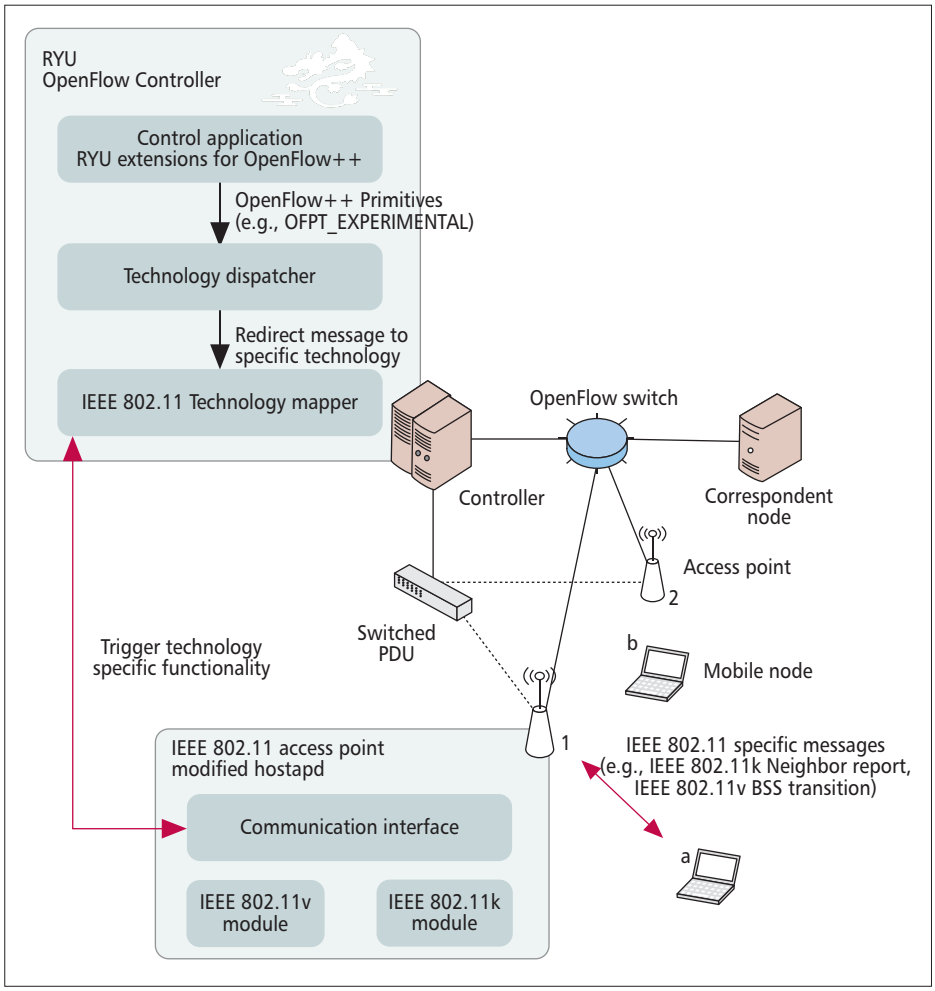

Figure 4. Deployed testbed and implemented modules and interfaces.

We also represent in Fig. 5 the estimated energy consumption of the infrastructure, following the model of [14]. Compared to the case of both APs on, our solution reduces energy consumption by 18 percent, an improvement that comes at the cost of increased delay, mainly because of 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 percent 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).

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 support for heterogeneous technologies by making use of abstractions, and enables the use of infrastructure-on-demand schemes via novel primitives to switch devices on/off as required. The validity of our approach has been demonstrated for the case of IEEE 802.11 through a simple prototype.

In addition to our standardization activities discussed above and the scalability analysis of the architecture described above, we are working on the following challenges:

- We are currently extending the prototype to Long Term Evolution (LTE)-like environments by adapting the open source LTE/EPC Network Simulator LENA[10] to run on top of a software defined radio.
- We are also deploying our framework in a campus-size

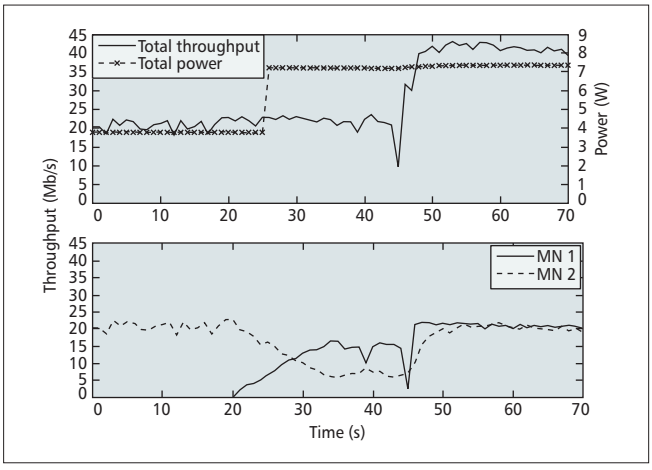

Figure 5. Validation of the proof of concept.

[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.

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

testbed to evaluate the performance improvements that can be achieved,[11] analyzing 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; http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[2] W. Sun *et al.*, "WiFi Could Be Much More," *IEEE Commun. Mag.*, vol. 52, no. 11, Nov. 2014, pp. 22–28.

[3] P. Serrano *et al.*, "Greening Wireless Communications: Status and Future Directions," *Comp. Commun.*, vol. 35, no. 14, 2012, pp. 1651–61.

[4] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comp. Commun. Rev.*, vol. 38, no. 2, Mar. 2008, pp. 69–74.

[5] C. Bernardos *et al.*, "An Architecture for Software Defined Wireless Networking," *IEEE Wireless Commun.*, vol. 21, no. 3, June 2014, pp. 52–61.

[6] Y. Yiakoumis *et al.*, "BeHop: A Testbed for Dense WiFi Networks," *Proc. 9th ACM Int'l. Wksp. Wireless Network Testbeds, Experimental Evaluation and Characterization '14*, 2014, pp. 1–8.

[7] L. Suresh *et al.*, "Towards Programmable Enterprise WLANS with Odin," *Proc. 1st Wksp. Hot Topics in Software Defined Networks '12*, 2012, pp. 115–20.

[8] X. Li, P. Djukic, and H. Zhang, "Zoning for Hierarchical Network Optimization In Software Defined Networks," *IEEE NOMS '14*, May 2014, pp. 1–8.

[9] X. Li and H. Zhang, "Creating Logical Zones for Hierarchical Traffic Engineering Optimization In SDN-Empowered 5G," *Proc. Int'l Conf. Computing, Networking and Commun.*, 2015, 2015.

[10] F. Ganji *et al.*, "Greening Campus WLANs: Energy-Relevant Usage and Mobility Patterns," *Computer Networks*, Special Issue on Green Communications, 2014; http://www.tkn.tu-berlin.de/fileadmin/fg112/Papers/2014/Ganji14greeningcampuswlans.pdf

[11] A. De La Oliva, A. Banchs, and P. Serrano, "Throughput and Energy-Aware Routing for 802.11 Based Mesh Networks," *Comp. Commun.*, vol. 35, no. 12, July 2012, pp. 1433–46.

[12] A. Garcia-Saavedra *et al.*, "SOLOR: Self-Optimizing WLANs with Legacy-Compatible Opportunistic Relays," *IEEE/ACM Trans. Networking.*, vol. PP, no. 99, 2014.

[13] I. F. Akyildiz *et al.*, "A Roadmap for Traffic Engineering In SDN-Openflow Networks," *Computer Networks*, vol. 71, no. 0, 2014, pp. 1–30.

[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

[14] P. Serrano *et al.*, "Per-frame Energy Consumption in 802.11 Devices and Its Implication on Modeling and Design," *IEEE/ACM Trans. Networking*, accepted for publication.

[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," *Proc. 4th IFIP Conf. Sustainable Internet and ICT for Sustainability*, Apr. 2015; http://www.it.uc3m.es/pablo/papers/impact-start-up-times.pdf.

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 Ph.D. while working at IMDEA Networks Institute. His research focuses on performance evaluation and centralized 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 journals 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 TPCs 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 Ph.D. in 2008 from UC3M, and has been working as associate professor there since then. His current line of research is related to networking in extremely dense networks. He is an active contributor in 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 a 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 has had a double affiliation as Deputy Director of IMDEA Networks Institute.

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 IETF. He has served as a Guest Editor of *IEEE Network*.