Intent–based Mobile Backhauling for 5G Networks

Tejas Subramanya, Roberto Riggio, and Tinku Rasheed
CREATE-NET, Trento, Italy; E–Mail: t.subramanya,roberto.riggio,tinku.rasheed@create-net.org

***Abstract*—Intent–based networking is a major component that will transform the manner in which the SDN/NFV–enabled future network infrastructures are operated. In particular, Intent-based networking is expected to play a major role in the multi–technological and software–defined 5G systems development roadmap. In this paper, we present the design and prototype implementation of an Intent–based mobile backhauling interface for 5G networks. Finally, we report on the empirical evaluation of of the proposed Intent–based interface over a small Enterprise WLAN. We also release the entire software stack under a permissive license for academic use.**

***Index Terms*—Wireless Networks, Network Function Virtualization, Intent–based Networking**

I. Introduction

Despite the fact that mobile networks are growing more complex over the years, the network management tools have not conceptually changed in the last decade. Software Defined Networking (SDN) has brought to the networking landscape concepts and paradigms that have been common for a long time in the computer science domain, such as high–level programming abstractions and declarative languages. At the same time, a new trend has recently emerged in the networking domain calling for a radical refactoring of network functions. This trend, known as Network Function Virtualization (NFV), points toward a transition from hardware based middleboxes to virtual network functions (VNF) running as software processes on general purpose platforms.

If, due to the improved economies of scale, this transition is set to reduce the deployment and operational costs of current networks, several implications of such software–centric networks are still unknown. New bugs, increased latency, and in general less predictable performances are just some of the pitfalls typically linked to NFV. Moreover, the requirements of a network service are often specified by business stakeholders and customers as high level policies. For example, the customer of an Infrastructure as a Service provider may be interested in a overlay network spanning all its worldwide branch offices. However, the same customer may also want assurance that its traffic is not routed across certain nodes. Similarly, a virtual wireless network operator may want all the traffic coming for its wireless users to be routed trough a certain node (e.g. a Performance Enhancement Proxy). Finally, recent advances in wireless communications, such as Coordinated Multipoint (CoMP), are also set to raise new network control and coordination challenges.

In this paper, we introduce a novel multi–domain/multi–technology Intent–based networking interface. Its design is based on the requirements of future 5G mobile networks which blur the line between radio access and backhaul in favor of a programmable data–plane combining computational and networking resources (including radio). Our design accounts for several scenarios including mobility management, uplink/-downlink decoupling, and fine grained packet processing. This paper also reports on a preliminary implementation of the proposed Intent–based interface and on its evaluation over an enterprise WLAN network. Preliminary results show that our design can ensure functional correctness of the requested network service while at the same time demonstrating good performance. We also release a proof–of–concept implementation of our intent–based networking interface under a permissive APACHE 2.0 License for academic use.

The rest of the paper is structured as follows. Section II discusses the related work. Section III introduces the requirements and the system design. The proof–of–concept and its validation are presented respectively in Sec IV and in Sec V, respectively. Finally, Sec. VI draws the conclusions pointing out our future research directions.

II. Related Work

Rapidly emerging technologies such as SDN, NFV, and Cloud computing are transforming Telco networks from a network–centric to an application–centric view. Intent–Based Networking is one such approach which allows network service developers to express their requirements and constraints in the form of policies, i.e. *what to do*, rather than the mechanism, i.e. *how to configure the network*. These directives, known as "Intents", are composed by network service developers with little or no knowledge about the underlying infrastructure. Eventually, intents are compiled into low–level forwarding rules by an Intent–decoder. Currently, researchers from academia, industry and open software communities are actively investigating the importance of the intent–based approach in the field of networking [1], [2], [3].

SDN and Cloud Computing Platforms. The OpenDayLight [4] project is developing a Network Intent Composition (NIC) framework as part of the OpenDaylight Controller platform [5]. OpenDayLight also supports two other types of intent, namely GBP–intent [6] and NEMO [7]. The Open Source Network Operating System (ONOS) project also provides an intent framework that allows applications to specify network control desires in terms of policies [8]. OpenStack [9], a popular open source cloud computing platform, also supports the GBP framework [10].

Network Policies. Many works have recently focused on flexible network policies [11], [12], [13]. These frameworks are however complex to use by non-expert users because

Research leading to these results received funding from the European Union's H2020 Research and Innovation Action under Grant Agreements H2020-ICT-671639 (COHERENT).

978-3-901882-85-2 © 2016 IFIP

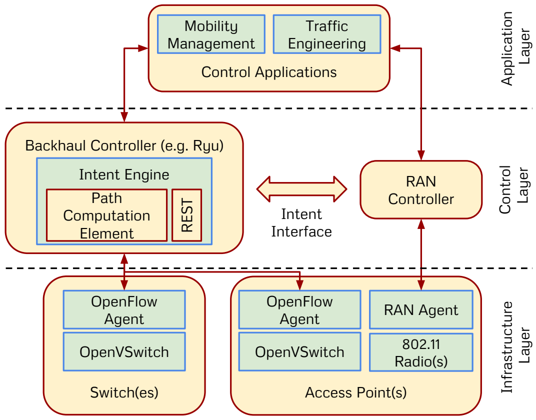


Fig. 1: System Architecture.

policy expressions are tied with low level details of the packet forwarding nodes. In [14], [15], the authors propose frameworks to manually compose network policies. In [16], [17], the authors argue that the manual composition of network policies by tenants, operators, network admins, end–users and control programs (network services) independently might result in network policy conflicts and unexpected runtime behavior. Therefore, they propose a graph based approach to resolve policy conflicts. In [18], the authors argue that currently there is no comprehensive network virtualization platform and that it is time to revise the network management state of the art. Therefore, they propose an intent-based modeling abstraction for specifying the network as a policy and also present an efficient network virtualization platform called DOVE to realize their proposed abstractions.

Nevertheless, to the best of our knowledge, this is the first work to propose an intent–based interface addressing the requirements of a programmable mobile backhaul capable of supporting the advanced features that will characterize 5G systems, including multiple points of attachment in both the uplink and downlink directions, flexible functional split, and fine grained packet control.

III. Intent–based Networking Abstractions

A. Requirements

In this section we set to define a reference model for Intent–based networking in a multi–domain/multi–technology scenario combining radio access and backhauling. Figure 1 depicts the high–level reference system architecture. As it can be seen, it consists of three layers: infrastructure, control and application. The infrastructure layer includes the data–plane network elements (e.g. Wi–Fi Access Points and OpenFlow switches) which are in constant communication with the (logically) centralized controller(s) situated at the control layer. Applications run in the application layer leveraging on the global network view exposed by the controller(s) to implement the network control and coordination tasks. The backhaul is assumed to be OpenFlow–enabled [19].

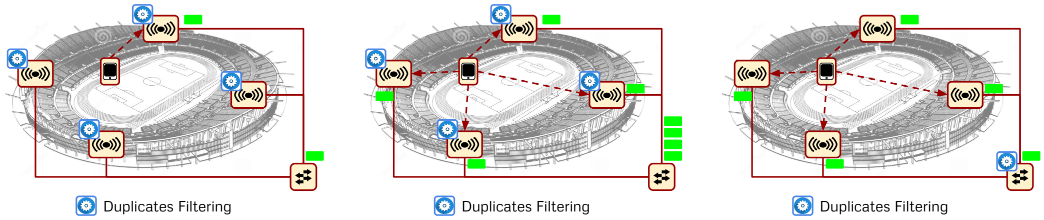
Let us consider a WLAN deployed in a football stadium such as the one depicted in Fig. 2a. Notice that, this work is agnostic w.r.t. the specific radio access technology, consequently the use of a Wi–Fi based WLAN in this section is to be considered merely as an example and not as a system requirement. Due to both the high number of potential wireless clients (football stadiums can often accommodate more than 100000 people) and the contention–based nature of the Wi–Fi MAC, the quality of experience perceived by the users can quickly degrade as more and more user–generated content is injected into the network (e.g. tweets and vines). However, although in a typical Wi–Fi network clients are connected to a single AP which acts as their point of attachment to the wired backhaul (see Fig. 2a), the broadcast nature of the wireless medium allows, in principle, to opportunistically receive a wireless user's transmission at multiple in–range APs (see Fig. 2b). If the collisions are i.i.d., then the overall frame delivery probability will increase with the number of APs that are within decoding range of the transmitting user. However, a naive implementation of this mechanism will inevitably generate a high number of duplicates on the wired backhaul. This is detrimental to the performance of both the backhaul, due to the increased traffic, and of TCP, because duplicate segments are interpreted by TCP as a sign of congestion.

Off–the–shelf Wi–Fi APs already implement local duplicates filtering, since duplicate frames can be received in case of a lost acknowledgment. In Fig. 2c instead, the duplicate filtering functionality which is typically embedded within the Wi–Fi APs is decoupled and moved to the backhaul. Duplicate filtering could be performed by either dedicated servers deployed in the backhaul or it could be performed by hybrid switching nodes embedding packet processing capabilities. In either case the duplicate filtering operation can be seen as VNF executed on a general purpose, possibly virtualized platform. Finally, a the stadium infrastructure owner could require that web traffic generated by the users must be sent to a Deep Packet Inspection VNF while the remaining traffic, e.g. emails, can be forwarded directly to the global Internet. The requirements imposed by this use case on the backhaul Intent–based interface are the following:

- *Dynamic chaining*: precise portions of wireless traffic must be processed by a set of VNFs in a pre–defined order; no knowledge of the underlying substrate network shall be required on the Intent interface consumer.
- *Multiple Points of Attachment*: wireless clients can have multiple points of attachment to the wired backhaul; traffic for a single destination may thus be required to be routed to/from different point of attachment.
- *Mobility Management*: wireless users' points of attachment can change at run–time due to user mobility; the intent interface consumer shall be allowed to declare the actual point of attachment(s) leaving to the runtime system the burden of re–configuring the backhaul.

B. Design

The requirements listed above are captured by the Intent–based mobile backhauling abstractions depicted in Fig. 3 as a UML class diagram. As it can be seen, an Intent is defined as a collection of *Virtual Links*. Each *Virtual Link* represents a



(a) Single uplink.

(b) Multiple uplinks, no duplicates filtering.

(c) Multiple uplinks, with duplicates filtering.

Fig. 2: A Wi–Fi WLAN deployed in a football stadium. In (a) transmissions from wireless users are received at a single AP while in (b) transmissions are opportunistically received at multiple APs. Duplicates are filtered in the backhaul (c).

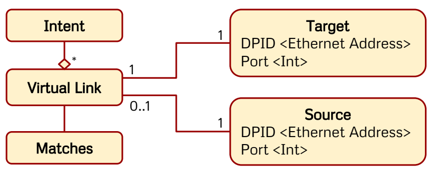


Fig. 3: Intent–based mobile backhauling abstractions.

backhaul forwarding policy. *Virtual Link*s must specify at least the *Target Termination Point* (TTP) and the portion of the flow space to which the Intent shall be applied. An optional *Source Termination Point* (STP) can also be specified. A *Virtual Link* with just the TTP requires the backhaul to forward all the matching flows to the TTP. Conversely, a *Virtual Link* with both the TTP and the STP requires the backhaul to forward all the matching flows at the STP to the TTP. Termination points are defined as the pair Datapath Identifier (DPID) / Port. The portion of the flow space affected by the intent is instead defined as a standard OpenFlow matching rule.

For example, let us consider the scenario sketched in Fig. 4. The network is composed of three Wi–Fi APs and two OpenFLow switches. The single wireless user has one point of attachment in the downlink direction and three points of attachment in the uplink direction. One duplicate filtering VNF is available in the network attached to one of the OpenFlow switches. The Intent configuration required by this scenario is composed of four *Virtual Link*s. The first *Virtual Link* requires the backhaul to deliver all the traffic destined to the wireless station *AA:BB:CC:DD:EE:FF* to DPID *00:00:00:00:00:01* port *1*. The remaining *Virtual Link*s require the backhaul network to deliver all the traffic transmitted by the wireless station *AA:BB:CC:DD:EE:FF* and received by the three APs, to the *00:00:00:00:00:0A* port *4*. The latter is essentially the port to which the Duplicate filtering VNFs is attached. The *Virtual Link*s are depicted as dashed blue lines in Fig. 4.

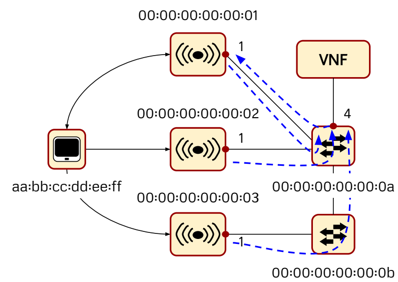


Fig. 4: A sample backhaul network. Virtual links are represented as dashed blue lines.

IV. Implementation Details

A. Wireless Access Platform

The proposed Intent–based networking interface has been implemented and validated using the EmPOWER platform. EmPOWER is an open toolkit for SDN/NFV research and experimentation in wireless and mobile networks. Its flexible architecture and the high–level programming APIs allow for fast prototyping of novel services and applications. EmPOWER relies on a centralized controller to implement control and management tasks. EmPOWER currently supports both Wi–Fi and LTE radio access nodes [20]. EmPOWER also supports NFV Function Management and Orchestration for Click–based [21] Light Virtual Network Functions [22].

B. Intent Engine

As OpenFlow controller for the wired backhaul we have selected the Ryu [23]. Notice however, that although the proof–of–concept presented in this work targets he Ryu Controller, the Intent interface design itself is controller agnostic. We extended Ryu by implementing a new component named *Intent Engine*. Such component is in charge of both receiving new Intents and translating them into actual rules for the backhaul. The *Intent Engine* is composed of two parts: a REST interface and a Path Computation Element. Intents are received from the REST interface as series of POST commands, one for each *Virtual Link* in the Intent. For each valid *Virtual Link*, the Path Computation Engine generates a set of FlowMod commands implementing the requested *Virtual Link* forwarding policy.

The listings below contains the REST requests implementing the intent depicted in Fig. 4. In particular the following REST request contains the downlink *Virtual Link*. As it can be seen, the *Virtual Link* is requiring the backhaul network controller to forward all the traffic addressed to station *aa:bb:cc:dd:ee:ff* to DPID *00:00:00:00:00:01* on port *2*.

Listing 1: Basic mobility management Intent

```
{
  "ttp_dpid": "00:00:00:00:00:01",
  "ttp_port": 2
  "matches": {
    "dl_dst": "aa:bb:cc:dd:ee:ff"
  }
}
```

Instead, the next listing contains the REST request for the uplink *Virtual Link*s. In this case the *Virtual Link* is requiring the backhaul network controller to forward all the traffic with the specified pairs of Ethernet source/destination addresses received on DPID *00:00:00:00:00:01* on port *2* to the DPID *00:00:00:00:00:0A* on port *4*.

Listing 2: Duplicate filtering Intent.

```
{
  "stp_dpid": "00:00:00:00:00:01",
  "stp_port": 1
  "ttp_dpid": "00:00:00:00:00:0A",
  "ttp_port": 4
  "matches": {
    "dl_src": "AA:BB:CC:DD:EE:FF"
    "dl_dst": "5C:E0:C5:AC:B4:A3"
  }
}
```

Notice how, during the serialization, the *Virtual Link* definition has been slightly changed. This is due to the fact that, in order to properly operate, the Duplicate Filtering VNF must have access to the sequence number in the Wi–Fi header. However, Wi–Fi frames cannot be directly transported over the backhaul, since their format would not be recognized by the OpenFlow switches. Instead, before entering the wired backhaul, Wi–Fi frames must be first encapsulated into a suitable transport protocol such as the Lightweight Access Point Protocol (LWAPP) [24]. LWAPP frames can then be carried over Ethernet. In our prototype LWAPP frames source Ethernet Address is set to the MAC Address of the wireless client, while the destination address is set to the MAC address of the interface to which the LWAPP message is sent. This process of encapsulation is transparently performed by the AP. That is why the *Virtual Link* requires the backhaul network controller to match on the Ethernet source *AA:BB:CC:DD:EE:FF* and on the Ethernet destination address *5C:E0:C5:AC:B4:A3* which we assume to be the VNF interface MAC address.

C. Duplicate Filtering

The duplicate filtering VNF is implemented using Click [21] on a standard laptop attached to one of the backhaul switches. The Click configuration is reported following listing.

Listing 3: Duplicates filtering VNF.

```
FromDevice(eth0)
 -> Classifier(12/88bb)
 -> Strip(18)
 -> WifiDupeFilter()
 -> WifiDecap()
 -> ToDevice(eth0)
```

As it can be seen non–LWAPP traffic is discarded, while legit frames are stripped of their Ethernet (14 bytes) and LWAPP (4 bytes) headers and passed to the duplicate filtering element (*WiFiDupeFilter*). Unique WiFi frames are then converted to Ethernet frames and sent back to the backhaul.

V. Evaluation

A. Methodology

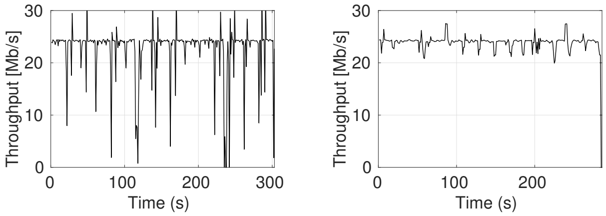
The main goal of our experiments is to determine the benefits of Intent–based networking on Mobility Management. The experimental setup is the one depicted in Fig. 4. The wireless access network and the backhaul network controllers (not represented in the figure) communicate with the networking elements using dedicated Ethernet links (out–of–band signaling). During the measurement we instructed the WLAN controller to periodically hand–over the single wireless station to a different APs. Details about how Wi–Fi handovers are implemented can be found in [20].

Measurements are taken in the downstream and upstream directions using TCP traffic with and without the Intent–based interface. In the latter case, the backhaul reconfiguration is left to the learning switch algorithm implemented by the OpenFlow switches. Iperf [25] was used as synthetic traffic generator. Handovers are performed every 5 seconds. Each measurement was 300 seconds long.

B. Measurement results

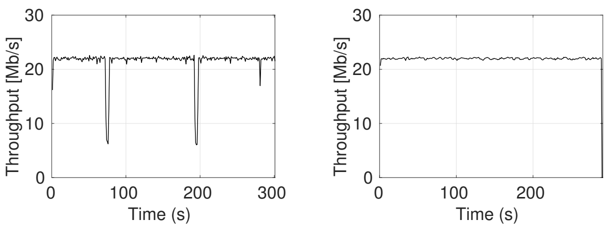
Results for both the downstream and upstream traffic are reported in, respectively, Fig. 5 and in Fig. 6. As it can be seen the introduction of the Intent–based interface provides some benefits. The performance improvement is particularly noticeable in the upstream direction where the use of the Intent–based interface essentially eliminates any performance degradation due to the handovers. This is to be ascribed to the fact that in the upstream scenario the wireless client is uploading a significant amount of data to a remote server. However, in the legacy scenario TCP acknowledgments may be delivered to the old AP when a handover happens. This is interpreted by TCP as a sign of congestion triggering a reduction in the transmission rate at the sender.

Conversely, in the downstream direction the performance improvement is not as remarkable. In fact, a non–negligible performance degradation can be noticed when handovers happen. This degradation is nevertheless smaller than the one experienced when no intent–based interface is used. The difference in behavior can be ascribed to the higher data–rate in the downstream direction which means that some TCP segments may still be on route to the old AP a handover happens. Finally, Fig. 7 plots the TCP throughput (samples are taken every 1 second) distribution with and without Intent–based interface. The experiments was also performed using UDP traffic (single CBR flow at 20 Mb/s), however in this case the performance improvement brought by the intent–based interface was essentially null.



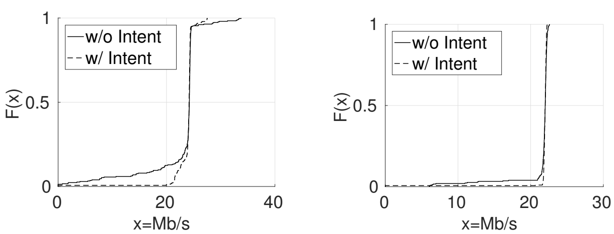
(a) w/o Intent–based Interface. (b) w/ Intent–based Interface

Fig. 5: TCP throughput in the downstream direction with and without Intent–based interface.



(a) w/o Intent–based Interface. (b) w/ Intent–based Interface

Fig. 6: TCP throughput in the upstream direction with and without Intent–based interface.



(a) Downstream direction. (b) Upstream direction.

Fig. 7: TCP throughput distribution with and without Intent–based interface.

VI. Conclusions

In this paper we presented an Intent–based backhauling interface for 5G systems. The proposed interface draws a clear line between the concerns of the wired backhaul controller and the concerns of the wireless access controller. We have also reported on a preliminary proof–of–concept implementation of the proposed interface. Empirical results show significant performance improvements.

As future work we plan to extend the Intent interface with support for VNF migration, path restoration, and telemetry. Moreover, we also plan to extend the Intent interface with bidirectional communications allowing the backhaul controller to notify the wireless access controller about network failures and topology changes. Conflict resolution between policies is also left as future work. Finally, we also plan to port the interface to other platforms, such as ONOS [26].

References

[1] "Intent As The Common Interface to Network Resources." [Online]. Available: http://www.ietf.org/mail-archive/web/i2nsf/current/pdfEhAfL7kT9F.pdf

[2] "Intent: Don't Tell Me What to Do! (Tell Me What You Want)." [Online]. Available: https://www.sdxcentral.com/articles/contributed/network-intentsummitperspectivedavidlenrow/2015/02/

[3] "The Most Important Work in SDN: Have We Got It Backward?" [Online]. Available: https://www.sdxcentral.com/articles/contributed/important-work-sdn-got-backwards/2014/04/

[4] "The OpenDaylight Project." [Online]. Available: https://www.opendaylight.org/

[5] "OpenDayLight Network Intent Composition." [Online]. Available: https://wiki.opendaylight.org/view/Network_Intent_Composition Use_Cases

[6] "OpenDayLight Group Based Policy." [Online]. Available: https://wiki.opendaylight.org/view/Group_Based_Policy_(GBP)

[7] "NeMo: An Applications Interface to Intent Based Networks." [Online]. Available: http://nemo-project.net

[8] "ONOS Intent Framework." [Online]. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework

[9] "OpenStack Open Source Cloud Computing Software." [Online]. Available: https://www.openstack.org/

[10] "OpenStack Group Based Policy." [Online]. Available: https://wiki.openstack.org/wiki/GroupBasedPolicy

[11] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," *SIGPLAN Not.*, vol. 46, no. 9, pp. 279–291, Sep. 2011.

[12] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software-defined networks," in *Proc. of USENIX NSDI*, Lombard, IL, USA, 2013.

[13] A. Voellmy, J. Wang, Y. R. Yang, B. Ford, and P. Hudak, "Maple: Simplifying sdn programming using algorithmic policies," in *Proc. ACM SIGCOMM*, Hong Kong, China, 2013.

[14] C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger, and D. Walker, "Netkat: Semantic foundations for networks," *SIGPLAN Not.*, vol. 49, no. 1, pp. 113–126, Jan. 2014.

[15] N. Foster, D. Kozen, M. Milano, A. Silva, and L. Thompson, "A coalgebraic decision procedure for netkat," *SIGPLAN Not.*, vol. 50, no. 1, pp. 343–355, Jan. 2015.

[16] J. Lee, J.-M. Kang, C. Prakash, S. Banerjee, Y. Turner, A. Akella, C. Clark, Y. Ma, P. Sharma, and Y. Zhang, "Network policy whiteboarding and composition," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 373–374, 2015.

[17] C. Prankash, J. Lee, Y. Turner, J.-M. Kang, A. Akella, S. Benerjee, C. Clark, Y. Ma, P. Sharma, and Y. Zhang, "PGA: Using Graphs to Express and Automatically Reconcile Network Policies," in *Proc. of ACM SIGCOMM*, New York, NY, USA, 2015.

[18] R. Cohen, K. Barabash, B. Rochwerger, L. Schour, D. Crisan, R. Birke, C. Minkenberg, M. Gusat, R. Recio, and V. Jain, "An Intent-based Approach for Network Virtualization," in *Proc. of IFIP/IEEE IM*, Ghent, Belgium, 2013.

[19] 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.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[20] R. Riggio, M. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rasheed, "Programming abstractions for software-defined wireless networks," *Network and Service Management, IEEE Transactions on*, vol. 12, no. 2, pp. 146–162, June 2015.

[21] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, Aug. 2000.

[22] R. Riggio, I. G. B. Yahia, S. Latr, and T. Rasheed, "Scylla: A Language for Virtual Network Functions Orchestration in Enterprise WLANs," in *Proc. of IEEE NOMS*, Istanbul, Turkey, 2016.

[23] "Ryu." [Online]. Available: https://osrg.github.io/ryu/

[24] "Lightweight Access Point Protocol," Internet Requests for Comments, RFC Editor, RFC 5412, 2010. [Online]. Available: https://www.ietf.org/rfc/rfc5412.txt

[25] "Iperf." [Online]. Available: http://iperf.sourceforge.net/

[26] "Open Network Operating System (ONOS)." [Online]. Available: http://onosproject.org/