Integrating NFV-MANO with Wireless Services: Experiences and Testbed Development

Nikos Makris[⋆†], Christos Zarafetas[⋆], Alexandros Valantasis[⋆], Thanasis Korakis[⋆†] and Leandros Tassiulas[‡]
[⋆]Department of Electrical and Computer Engineering, University of Thessaly, Greece
[†]Centre for Research and Technology Hellas, CERTH, Greece
[‡]Department of Electrical Engineering, Yale University, New Haven, USA
Email: {nimakris, hrzarafe, valantas, korakis}@uth.gr, leandros.tassiulas@yale.edu

***Abstract*—Network Functions Virtualization Management and Orchestration (NFV-MANO) aims to provide a common interface for technology developers and service operators for the instantiation of virtual network functions over generic equipment. Nevertheless, although NFV-MANO is currently targeting datacenter operations and the deployment of virtual services (through Virtual Machines or containers), it can easily extend to generic networking devices, and alter their operation based on the deployed network functions. In this paper, we consider the case of a wireless testbed, with focus on wireless networking, and adopt the OpenSourceMANO framework for provisioning of services on top of the equipment. We provide the necessary extensions to the framework in order to deploy services over virtualized wireless network interfaces, hosted on the generic networking nodes of the testbed. The extensions we focus on regard the provisioning of virtual functions over wireless links, that are traditionally not handled by the framework, and allow easier interaction of the end-users with the testbed.**

***Index Terms*—OpenSourceMANO, NFV-MANO, wireless, testbed, NITOS**

I. Introduction

Network Functions Virtualization (NFV) is aiming at offering more agile resources to network providers, with improved sustainability due to the softwarized nature of the resources. The key features of NFV include reduced CAPEX and OPEX costs, optimized reconfiguration of the provided services, easy integration of new services and operational efficiency due to the concepts of network slicing and multi-tenancy (i.e. multiple operators using the same physical infrastructure). Due to the above reasons, NFV is considered to be one of the major enablers for the 5G technology [1]. Nevertheless, Management and Orchestration (MANO) of the underlying hardware resources is not an easy task, as the interplay of the heterogeneous underlying equipment, providing different APIs for their configuration, needs to be efficiently defined. To this aim, the NFV-MANO [2] working group was formed by ETSI with the purpose to define a generic architecture for the management and orchestration of virtualized resources. The hardware resources (compute, storage, network) are abstracted through the framework, whereas focus is placed on the efficient interconnection of the orchestrated components.

The NFV-MANO architecture is providing the necessary abstractions of the underlying hardware equipment, and concentrates only on the orchestration, provisioning and cross-interaction of the deployed functions, taking care of all the low level configurations for setting up end-to-end paths between the functions. Nevertheless, NFV-MANO is mainly addressing datacenter resources, with the networking being programmed through the SDN concept, whereas services are deployed using either virtual machines or light-weight containers. However, this approach can extend to generic networking devices, as long as they are organized in a distributed fashion, which includes other technologies (such as wireless) that are not currently addressed by SDN through production grade software.

In this paper, we present our approach in transforming an open wireless testbed to an NFV-MANO compliant platform. Taking advantage of the available wireless hardware supporting different technologies (LTE, WiFi, mmWave) we tailor the OpenSourceMANO (OSM) [3] platform, the most widely adopted framework for NFV-MANO, to provide services running on top of different wireless networks. Traditionally, tools like OSM do not deal with network connections other than Ethernet, and make use of network interface virtualization enablers such as SR-IOV. Nevertheless, similar solutions for the virtualization of the wireless interfaces also exist, and can be used in order to bind services that are orchestrated with these frameworks over wireless connections. The resulting ecosystem is a powerful platform allowing the orchestration of experiments using virtual resources, that off-the-shelf supports portability from any other OSM-compliant site to the testbed.

The rest of the paper is organized as follows: Section II is proving a brief description of the testbed's resources and former experimentation methodology. Section III is detailing our architecture for provisioning the testbed with the NFV-MANO paradigm. In section IV we discuss the experimentation potential of the platform, and in section V we conclude.

II. NITOS Testbed Environment

The target facility used to develop our scheme is the NITOS testbed (http://nitos.inf.uth.gr), located in University of Thessaly, Greece. The testbed is providing in a 24/7 fashion remotely accessible resources, targeting at wireless networking based experimentally driven research. The testbed is consisting of over 100 nodes, equipped with key technologies:

- All the nodes are high-end PCs, equipped with Core-i7 processors and 8 GBs of RAM each, and feature at least two IEEE 802.11 a/b/g/n/ac cards, compatible with Open Source drivers (e.g. ath9/10k) used for WiFi research.
- Two commercial off-the-shelf LTE access points are available for experimentation, along with the respective Core Network solution. Both femtocells and core network are programmable through testbed services that are available [4]. About half of the nodes are equipped with LTE

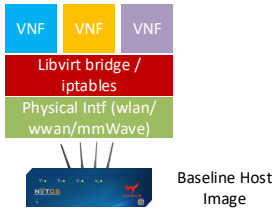


Fig. 1. VNF instantiation on a NITOS node: each VNF is either bridged/routed with iptables to the underlying physical wireless network (WiFi/LTE/mmWave).

dongles, that allow the establishment of an operator-grade LTE network, using testbed specific SIM cards.

- Over 20 different SDR devices exist in the testbed, that are compatible RF front-ends for open source implementations of base stations (such as OpenAirInterface [5]).
- Six mmWave devices are installed in the testbed, that are reachable from all the testbed's nodes and the creation of high-throughput wireless point-to-point links.
- All the nodes of the testbed are interconnected through different hardware OpenFlow switches, organized in a tree topology. Users can set their own controller to manage the flows of the nodes that they are using.

The testbed is organized in three different setups: An indoor RF-isolated, an outdoor setup prone to uncontrolled external interference and an office setup with mild interference settings. Resources can be mixed from the different locations in order to create a versatile experimentation environment. The nodes can be reserved through a portal service, for up to four hours per slot. Several tools are available for experimentation, supporting large scale effortless deployment of the environment (e.g. the cOntrol and Management Framework - OMF [6]).

III. System Architecture

The fact that traditionally NFV-MANO has been developed for orchestrating services and functions over datacenters hinders us from considering wireless services for deployment over them. Nevertheless, the structure of a testbed can be considered as a distributed set of nodes managed by an infrastructure manager such as Openstack or OpenVIM. For the differentiation of the services deployed over the nodes, we organize the testbed in four different "datacenters":

- **SDR datacenter**: all the nodes with SDR capabilities are included in this datacenter. VNFs/PNFs related to the execution of services interfacing SDRs are using this datacenter.
- **LTE datacenter**: all the nodes that have LTE dongles are included in this datacenter. All the services using the LTE network to transmit data are using this datacenter.
- **WiFi datacenter**: all the nodes with WiFi connections are included in this datacenter. Similar to the previous one, all the VNFs/PNFs deployed can use a WiFi connection.
- **Ethernet/mmWave datacenter**: the rest of the nodes that have Ethernet connections and use the mmWave equipment.

Depending on the type of services to be deployed, we target deploying the VNFs on top of the physical nodes as shown in Fig. 1. The VNFs (Virtual Machines with ready to provision services) are making use of a bridged Ethernet connection with the physical interface that transmits the data over the air. For the cases that a bridged connection is not feasible, we route the traffic with new routing and *iptables* rules.

The VIM of the testbed which has been extended is OpenVIM [7]. We make use of existing services for the configuration of the LTE equipment (*bscontrol/LTErf* [4]), or the automatic configuration of the *hostapd* instance for creating WiFi Access Points in the testbed. In order to host VNFs that use the same virtualized physical wireless connection, we introduce different slicing schemes that are managed by the same services, whenever a VNF Description (VNFD) is received from the OSM instance. In the following subsections, we initially describe the employed methods for slicing the wireless infrastructure, and subsequently provide details on how each node is virtualized and orchestrated with OSM.

A. LTE Network Slicing

Although there has been significant work on the RAN based slicing for LTE (e.g. [8], [9]), yet, no commercial products seem to integrate such functionality. For the NITOS case, we employ the off-the-shelf LTE infrastructure that is offered. Our high level target in slicing the network with a holistic approach (considering the RAN and Core Network as one entity) and to provide guarantees for different users on the usage of the network resources offered by the infrastructure. For this purpose, we exploit the notion of different Packet Data Networks (PDNs) to achieve this functionality, as follows: each PDN is realized as a separate tunnel for user plane traffic, running from the PDN-GW component of the Core Network (Evolved Packet Core - EPC) to each network UE. For the RAN case, the PDN is mapped to different Access Point Names (APNs). When a UE requests to connect to the LTE network, the RRC messaging exchange is containing the APN details that the UE wishes to connect to. These messages are exchanged between the UEs and the EPC. If the information is valid, and such a PDN is configured in the infrastructure, the UE is admitted to the network. The user plane data (data the UE is sending over the network) are then transferred from the base station to the EPC and vice-versa over dedicated GTP tunnels. The GTP tunnels are aggregated to a single interface on the core network side, per each PDN.

Each PDN is a separate broadcast domain for the network, and the UEs of the network use addresses belonging to this domain. Since all the clients of the LTE network are from the EPC's perspective interfaced by single tunnel interfaces per each PDN, we are able to isolate the flows of each PDN. Although UEs may be associated to the same base station, and their traffic is traversed through the same EPC, the operate in an isolated manner if they belong to different PDNs. Using this functionality, we can throttle the traffic that is exchanged over the network per each PDN from the EPC side; all the exchanged user traffic is traversing the EPC, even if a UE is trying to reach a nearby UE. This throttling is based on the maximum Uplink (UL) and Downlink (DL) traffic that the clients in each PDN exchange over the LTE network. An illustration of this slicing functionality is depicted in Fig. 2. In the illustration, the blue (PDN1) and the red (PDN2) clients

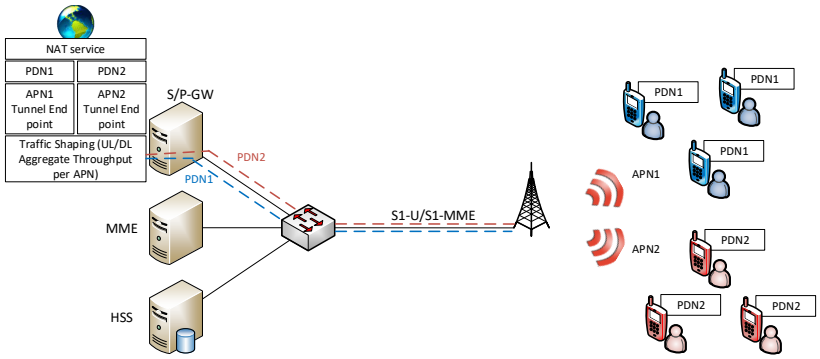


Fig. 2. LTE slicing based on PDNs in the NITOS testbed: each PDN equals to a unique path from the core network (P-GW component) to the wireless RAN, through which only clients belonging to the same PDN can communicate. Each PDN can be set to exchange a value of a maximum UL/DL aggregate traffic for all the clients belonging to it.

can only communicate with clients belonging to their own PDN. From the Core Network side, the traffic is throttled between the two different APNs, from the PDN-GW.

B. WiFi Network Slicing

For the WiFi case, we employ the *hostapd* service to setup wireless Access Points (APs). Through its configuration, *hostapd* can setup over a single physical card multiple AP instances, known as virtual Access Points (VAPs). Each VAP has separate settings for the transmitted ESSID parameter, which the end user devices employ to associate to the network. Physical network parameters, such as transmission power and channel configuration are the same across different VAPs. However, by using the separate data queues for the different types of data (Voice, Video, Best-Effort and Background) that the WiFi drivers have, prioritization between different VAP instances can take place, as it has been shown in other works, e.g. [10], [11]. However, given the Listen-Before-Talk (LBT) protocol of WiFi, all the end-users compete in order to get access to the medium. Hence, contrary to the LTE case, no guarantees can be given through these methods on the per-VAP exchanged traffic. Nevertheless, these methods regulate the probabilities of each VAP to access the channel in a percentage of the total available capacity, constrained by the external interference that is present in the served region.

C. mmWave Node Slicing

For the case of the mmWave nodes employed in NITOS we use the following approach. Each of the nodes is terminated to an OpenFlow switch, communicating over a specific VLAN. Different VLANs are allocated for each of the nodes. The termination switch of the mmWave nodes is connected to an other set of OpenFlow switches that interconnect the rest of the nodes of the testbed. All these switches can be programmed with individual controllers that each experimenter can setup, by slicing the switches using FlowVisor. For each user controller, only the ports that communicate with the reserved testbed nodes in this testbed account are included in the configurable flow space of the end user. On the mmWave nodes, Open-vSwitch (OvS) is used to bridge the Ethernet VLAN interface with the actual radio interface of the node. This gives us the advantage that all the nodes of the testbed can use the mmWave network connections, by setting up the correct VLAN interface on the node and enabling the respective VLANs on the OpenFlow switches.

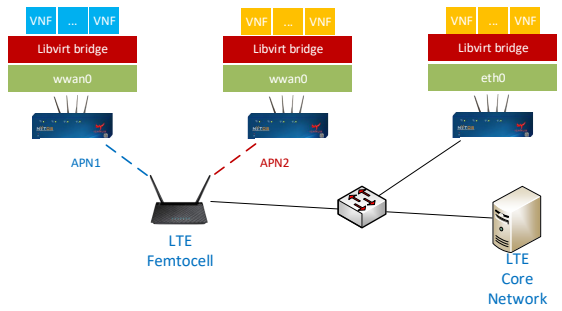
D. Testbed Virtualization Enablers

The instantiation process of a chain of VNFs has the following workflow. Initially the user interfaces a User Interface (UI), where the to be deployed VNFs are selected by a catalog of supported services. These services are subsequently configured in terms of their network interfaces (e.g. SR-IOV), and features regarding the virtualization technologies for the underlying physical processor, etc. The VNFs can be linked to form a chain (VNF Forwarding Graph - VNFFG), used to execute a specific process. When the user is instantiates the services, the VNF descriptions are sent to the underlying VIM for preparing and configuring the physical infrastructure that will host them.

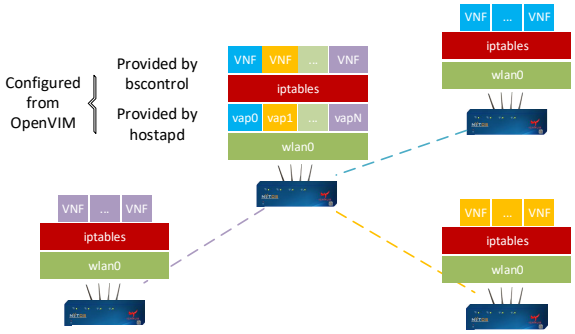
```
// Rest of VDU configuration omitted
  vdu:
  ...
  -
    name: "eth1"
    type: "EXTERNAL"
    external-connection-point-ref: "eth1"
    virtual-interface:
    type: "WIFI-ACCESS-POINT"
    vpci: "0000:00:0b.0"
    virtual-access-point:
   -
     id: "virtual-access-point-1"
     essid: "VWLAN1"
     mode: "g"
     channel: 1
     floating-ip-needed: "false"
// Rest of VDU configuration omitted
```

Listing 1. Sample of VDU configuration for a WiFi AP: multiple can be instantiated over the same node with the *bscontrol* service taking care of the low level networking and bridging configuration.

Before the instantiation of VNFs over the testbed's nodes (see Fig. 1), the aforementioned processes for slicing the infrastructure need to be invoked, so as the underlying physical connections are configured. In the traditional NFV-MANO architecture, these processes are configured by a network controller. For the case of the testbed, we extend the *bscontrol* service in order to enable such configuration. Hence, when the VNFDs are delivered from the MANO orchestrator to the VIM component of the testbed, the VIM is invoking specific commands of the *bscontrol* service. The service has a REST API, and is translating every incoming request to a string of



(a) Orchestration of VNFs belonging to specific slices of the LTE infrastructure. VNFs are orchestrated through OSM, whereas the LTE physical interconnection and bridging with the VNFs is offered through the *bscontrol* service. VNFs are able to communicate only on their PDN (or APN on the RAN side).



(b) VNF instantiation over WiFi links in the testbed: VAP creation and physical interconnection is provided by the *bscontrol service* whereas the *iptables* rules on the host controlling the VAP instances ensure that VNFs on the clients can communicate only with VNFs belonging to the specific VAP they are associated to.

Fig. 3. VNF Orchestration with WiFi and LTE connections.

commands that need to be executed over the host nodes of the infrastructure. For example, consider the case when the user wants to orchestrate a VNF that will be using a WiFi AP connection. When this type of connection is selected from the user for the VNF, the VNFD is augmented by a new section regarding the network configuration. This includes configuration of the wireless channel, the transmission power, the ESSID and the WiFi mode (a/b/g/n). When this information is extracted from the VNFD by the VIM, the *bscontrol* service is invoked, with the parameters being passed to it being the node that it will use as a host (which is outputted from the scheduler interface of the VIM) and the WiFi configuration.

```
// Rest of VDU configuration omitted
  vdu:
  ...
  -
    name: "eth1"
    type: "EXTERNAL"
    external-connection-point-ref: "eth1"
    virtual-interface:
     type: "LTE"
     vpci: "0000:00:0b.0"
     virtual-apn:
     -
      id: "virtual-apn-1"
      name: "APN1"
      qci: 5
      ulambr: 50000000
      dlambr: 50000000
// Rest of VDU configuration omitted
```

Listing 2. Sample of VDU configuration for a LTE AP: the client that will be attached to APN1 will get at max 50 Mbps for DL and UL traffic

The *bscontrol* service will subsequently access the node, via a secure shell connection, and setup the WiFi connection: 1) initially, it brings up the wireless driver, 2) configures the *hostapd* interface with multiple VAPs, 3) brings up the VAP and configures addressing on the physical interface and 4) sets up the forwarding rules for each VNF that will be deployed over the VAP. Subsequently, the VIM is deploying the VNF on the host, whereas all the underlying configuration has been taken care of. A similar procedure can take place for several VNFs that can be configured as WiFi Stations (STA) of specific VAPs. When this process is finished, and orchestration has taken place, we can have a topology that is illustrated in Fig. 3b. We note here that for the WiFi case, forwarding of the traffic of VNFs over the physical interfaces is not handled as the rest of the connections (e.g. using the *libvirt* bridge interface) but is using *iptables*. This comes from the fact that such wireless interfaces cannot be added to a bridge interface, as the bridge is provided by the *bridge-utils* package. This process is overcame by using an Open-vSwitch OpenFlow enabled bridge, but further configuration needs to take place for handling the separate flows per each VNF per each sub interface per each host machine. Therefore, *iptables* rules are established by the *bscontrol* service to setup the forwarding plane per each VNF on each host machine.

```
// Rest of VDU configuration omitted
  vdu:
  ...
  -
   name: "eth1"
   type: "EXTERNAL"
   external-connection-point-ref: "eth1"
   virtual-interface:
    type: "MM-WAVE"
    vpci: "0000:00:0b.0"
    millimeter-wave:
    -
     id: "millimeter-wave-2"
     physical-node-id: 2
     mcs: 7
// Rest of VDU configuration omitted
```

Listing 3. Sample of VDU configuration for a mmWave node: all the underlying VLAN configuration is handled by the NITOS *bscontrol* tool

For the LTE case, the VNFD is augmented with the PDN connection settings. These include the PDN configuration, the UL/DL Aggregate Bandwidth that will get over, and the subscriber information of the LTE dongle mounted on the host machine that VIM has scheduled to instantiate the VNF. Upon receiving this information, the *bscontrol* service configures the EPC with the PDNs, adds the subscriber to the PDN, and connects on the node and connects the UE to the LTE network of the testbed. In the case that a user orchestrates an Ethernet based client to use the PDN information of the LTE network, the service is additionally configuring up the routing rules for

the interconnection of the two hosts (see Fig. 3a).

From the aforementioned information, it is clear that the end-user only interfaces with the high level interconnection of the VNFs, and the network connection type over which the traffic will be traversed. The rest of the processes are taking place in a fully automated manner, and create the circumstances for the further integration and evaluation of other NFV-MANO functions to the testbed. Listings 1, 2 and 3 present the augmented YANG description of the VDUs used to orchestrate the WiFi AP and LTE nodes of the testbed. As we see, information about the wireless channel and name of the WiFi network are passed to the orchestrator for the WiFi case. Similarly, the APN configuration is being passed for the LTE case. This information is further redacted from the VIM, and by parsing it sends the appropriate commands to the bscontrol service that allows the configuration of the underlying networks. Once this process is done, the rest of the orchestration takes place, involving the deployment of services on top of the network bridges with the wireless connections.

IV. Discussion

The provided functionality is able to accelerate the deployment of new services that may be running over different networks, towards their experimentally driven validation. The presented approach, allows us to quickly and effortlessly deploy new services that are described in the format of VNFDs directly over the testbed, or any other physical infrastructure orchestrated through OSM. Hence, this approach is able to provide enhanced portability of experiments over the testbeds, regardless of their type, as long as they can process YANG based service descriptions. Providing such extensions to the testbed also allows us to milden the learning curve of using the infrastructure; the end-user is only presented with a repository of VNFs, which can be instantiated from a higher layer by organizing and designing the interconnection of the components through OSM. After the instantiation of the services, users get root access on the deployed VNFs.

Moreover, the service orchestration can be further implemented using tools like JuJu, Ansible or Cloud-Init, for the seamless and effortless bootstrapping and on-boarding of services on the VNFs. Processes that are based on software and usually take up a lot of time for the setup of the experimentation environment in the testbed, e.g. setting up a software based base station over a USRP device, can be fully automated, by the selection of the interfaces that will backhaul/fronthaul the base station from the top level.

Finally, a concern that arises from the integration of the YANG based description of services, is the intercommunication with the existing protocols for testbed resource reservation and access. The existing mechanisms are based on the Slice-based Federation Architecture (SFA) protocol [12], which represents the resources of the testbeds with XML documents, referred to as Resource Specifications (RSpecs). In this context, we consider the interplay between the two different formats of resources as follows: the users submit a VNF description of the services that is translated through testbed tools to physical resources (RSpecs), through appropriate calls to the SFA API of the testbed. After this process, the VNFs can be instantiated as normally using tools like OSM.

V. Conclusion

In this paper we presented some extensions to the well-known OSM framework for orchestrating VNFs over wireless links over the generic infrastructure in the NITOS wireless testbed. We presented the scheme for slicing the different wireless networks available in the testbed (LTE, WiFi and mmWave) and how this information has been propagated in the VNF descriptions. Dedicated testbed services are in charge of parsing this information (inside the VIM of OSM) and setting up the physical network interconnection between testbed nodes prior to the VNF deployment. In the future we foresee to integrate our extensions with other VIMs such as Openstack, and support newer versions of OSM (the existing functionality is tested in OSM Release TWO and THREE). Also, we aspire in creating virtualized instances of the software base stations, with automatic selection of the backhaul/fronthaul network interfaces from the orchestrator pane.

Acknowledgment

The research leading to these results has received funding by GSRT, under the act of "HELIX-National Infrastructures for Research", MIS No 5002781 and through the European Horizon 2020 Programme under grant agreement No 732497.

References

[1] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, "NFV and SDN-Key Technology Enablers for 5G Networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2468–2478, 2017.

[2] M. Ersue, "ETSI NFV management and orchestration-An overview," in *Proc. of 88th IETF meeting*, 2013.

[3] ETSI, "Open Source MANO," *OSM home page*, 2016.

[4] N. Makris, C. Zarafetas, S. Kechagias, T. Korakis, I. Seskar, and L. Tassiulas, "Enabling open access to LTE network components; the NITOS testbed paradigm," in *2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–6.

[5] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "OpenAirInterface: A flexible platform for 5G research," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 33–38, 2014.

[6] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "OMF: a control and management framework for networking testbeds," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4, pp. 54–59, 2010.

[7] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latré, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, 2016.

[8] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "CellSlice: Cellular wireless resource slicing for active RAN sharing," in *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 2013, pp. 1–10.

[9] M. I. Kamel, L. B. Le, and A. Girard, "LTE wireless network virtualization: Dynamic slicing via flexible scheduling," in *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*. IEEE, 2014, pp. 1–5.

[10] K. Kousias, K. Katsalis, D. Stavropoulos, T. Korakis, and L. Tassiulas, "Building virtual 802.11 testbeds towards open 5G experimentation," in *Wireless Communications and Networking Conference (WCNC), 2016 IEEE*. IEEE, 2016, pp. 1–7.

[11] Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. McKeown, "Slicing Home Networks," in *Proceedings of the 2nd ACM SIGCOMM workshop on Home networks*. ACM, 2011, pp. 1–6.

[12] L. Peterson, R. Ricci, A. Falk, and J. Chase, "Slice-based Federation Architecture (SFA)," *Working draft, version*, vol. 2, 2010.