Scheduling Wireless Virtual Networks Functions

Roberto Riggio, Abbas Bradai, Davit Harutyunyan, Tinku Rasheed, Toufik Ahmed

Scheduling Wireless Virtual Networks Functions

Roberto Riggio*, Abbas Bradai§, Davit Harutyunyan*, Tinku Rasheed*, Toufik Ahmed¶
*CREATE-NET, Trento, Italy; Email: rriggio,dharutyunyan,trasheed@create-net.org
§XLIM Institute, University of Poitiers, Poitiers, France; Email: abbas.bradai@univ-poitiers.fr
¶CNRS-LaBRI, University of Bordeaux, Bordeaux, France; Email: tad@labri.fr

***Abstract*—Network Function Virtualization (NFV) sits firmly on the networking evolutionary path. By migrating network functions from dedicated devices to general purpose computing platforms, NFV can help reducing the cost to deploy and operate large IT infrastructures. In particular NFV is expected to play a pivotal role in mobile networks where significant cost reductions can obtained by dynamically deploying and scaling Virtual Network Functions (VNFs) in the core and access segments. However, in order to achieve its full potential, NFV needs to extend its reach also the radio access network segment. Here Mobile Virtual Network Operators shall be allowed to request radio access VNFs with custom resource allocation solutions. Such requirement raises several challenges in terms of performance isolation and resource provisioning. In this work, we formalize the wireless VNF placement problem as an integer linear programming problem and we propose a VNF Placement heuristic named *WiNE* (Wireless Network Embedding) to solve the problem. Moreover, we also present a proof–of–concept implementation of an NFV management and orchestration framework for enterprise WLANs. The proposed architecture builds upon a programmable network fabric where pure forwarding nodes are mixed with radio and packet processing capable nodes.**

***Index Terms*—Network Management, Resource allocation, Virtual Network Embedding, Virtual Network Functions Placement, Network Function Virtualization, Wireless Networks.**

I. Introduction

Network Function Virtualization (NFV) promises to reduce the cost to deploy and operate large networks by migrating network functions from dedicated hardware appliances to software instances running on general purpose virtualized networking and computing infrastructures. This, in time, shall improve the flexibility and the scalability of mobile networks in that the deployment of new applications and services will be quicker (software vs hardware development life–cycles) and different network functions can share the same resources paving the way to further economies of scale.

This progressive process of network softwarization is set to play a pivotal role in the fifth generation of the mobile network architecture. In this context the Network–as–a–Service business model shall allow operators to tap into new revenue streams by further abstracting the physical network into service specific slices possibly operated by different mobile virtual network operators (MVNOs). The envisioned vertical applications range from high–definition video delivery to machine–to–machine applications and e–health.

In order to cope with the diverse range of requirements that sprout for such use cases, future wireless and mobile networks will further rely on virtualized resources and on dynamic service orchestration. Although a rich body of literature exists in the VNF placement [1], virtual network embedding [2], and component placement domains [3], most of these works focus on the problem of mapping an input virtual network request (often in the form of a VNF forwarding graph) onto a physical virtualized network substrate (often offering computational as well as networking resources). However, these works implicitly assume that once a VNF is mapped on a node, the network substrate virtualization layer will take care of scheduling the various VNFs ensuring both logical isolation and an efficient use of the substrate resources [4]. Such an assumption does not hold anymore if radio nodes are added to the set of virtualized resources available in the substrate network (alongside computational and networking resources). In this case, in fact, the amount of resources available at each substrate radio node is stochastic quantity depending on both channel fluctuations *and* end–users distribution.

In this work, we investigate the VNF placement *and* scheduling problems in the Radio Access Network (RAN) domain. In this scenario we expect MVNOs to specify their requests in terms of a VNF forwarding graph. Such VNFs can include functions such as load–balancing and firewall, as well as virtual radio nodes. Moreover, in order to satisfy the diverse requirements imposed by future applications and services, MVNOs must be allowed to deploy custom resource allocation schemes *within* their network slice. At the same time, the underlying system shall both enforce strict performance isolation between MVNOs and ensure efficient resource utilization across the network in spite of the non–deterministic nature of the wireless medium.

The contribution of this paper is twofold: (i) we formalize the VNF placement problem for radio access networks, and (ii) we propose a slice scheduling mechanism that ensures resource and performance isolation between different slices. The proposed solutions work jointly, i.e. performance isolation is ensured if slices are accepted under the constraints imposed by our VNF placement problem formulation. This paper extends our previous work [5] by refining the VNF placement heuristic *WiNE* (Wireless Network Embedding), extending the simulation study to additional types of VNF requests, and analyzing in deeper detail how the type of VNFs impacts on the substrate network utilization. Moreover, we also report on a updated proof–of–concept implementation of the proposed solution and on its field evaluation. The programmable data–path, the controller and the SDK have been released under a permissive license for academic use[1].

The rest of this paper is structured as follows. In Sec. II we discuss the related work. The physical network model,

[1]On–line resources available at: http://empower.create-net.org/

the VNF request model, and ILP problem formulation are presented in Sec. III. The VNF placement heuristics and its evaluation are presented in Sec IV and in Sec. V. The proof–of–concept is presented in Sec VI while some illustrative VNFs and their evaluation are presented in Sec. VII. Finally, Sec. VIII draws the conclusions pointing out the future work.

II. Related Work

The recent advances in general purpose computing platforms paved the way to a new generation of software routers. However, many of these solutions focus on improving the pure raw packet processing speed [6], [7], [8] but do not tackle the problem of deploying and orchestrating VNFs. In parallel, there are significant efforts toward VNF management and orchestration. In particular the European Telecommunications Standards Institute (ETSI) has recently tackled the NFV concept [9] while OPNFV [10], MANO [11], and OpenBATON [12] are working toward an open source carrier grade platform for NFV.

A. Virtual Network Embedding

The amount of literature on virtual network embedding (VNE) topic is considerable. Seminal works in this domain include VINEYard [13] for single domain VNE and PolyVINE [14] for multi–domain VNE. For a comprehensive survey on VNE algorithms we point the reader to [2]. In [15] the authors put forward a novel model that reflects the time–varying resource requirements of a virtual network request. The authors also consider virtual network embedding in the context of opportunistic resource sharing at the level of the entire substrate network. In [16] the authors present the *SiMPLE* virtual network embedding algorithm. *SiMPLE* exploits path diversity in order to protect virtual networks from single link failures. However, to the best of the authors knowledge, none of these works formulate the VNE problem for hybrid wired/wireless networks with the goal of ensuring performance isolation between tenants.

B. VNF Placement

The VNF placement problem is conceptually similar to component placement in data–centers and clouds. The amount of literature in this domain is thus humbling [17], [18], [19], [4]. A survey on resource management in cloud computing environments can be found in [3]. In [17] the authors study the problem of placing virtual machine instances on physical containers in such a way to reduce communication overhead and latency. In [18] the author propose a novel design for a scalable hierarchical application components placement for cloud resource allocation. The proposed solution operates in a distributed fashion, ensuring scalability, while providing performances very close to that of a centralized algorithm. This work is extended in [19] where several algorithms for efficient data management of component-based applications in cloud environments are proposed. In [4] the elasticity overhead and the trade–off between bandwidth and host resource consumption are jointly considered by the authors when formulating the VNF placement problem. In [20] a joint node and link mapping algorithm is proposed. While the authors of [21], [1], [22] tackle the problem of dynamic VNF placement. Targeting resource allocation in data–centers, these works do not tackle the problem of virtualized radio function placement.

C. Wireless & Mobile Networks

The topic of radio resources virtualization has received significant attention in the literature. In [23], a WLAN virtualization approach named *Virtual WiFi* is proposed extending the virtual network embedding from the wired to the wireless domain. Kernel–based virtual machines are used as a virtual wireless LAN devices. Time domain multiplexing is used in order to provide isolation between the virtual wireless devices. In [24], [25], wireless network virtualization is applied to wireless mesh networks. A virtual network traffic shaper is introduced in [26], [27] for air time fairness in 802.16e networks. In [28], [29] the problem of virtualizing OFDMA–based wireless networks (i.e. WiMAX and LTE) is studied. The authors tackle the problem both at the radio and the core network level opening the way to interesting infrastructure sharing scenarios. Similar consideration can be also made for [30] where a framework for sharing a single WiMAX base station is proposed. Wireless Virtualization of 802.11 devices is the focus of [31]. In all the cases above, however, the channel–aware placement of VNFs over *radio* and *wired* resources is not formulated nor is the performance isolation challenge between multiple MVNOs tackled.

D. Middlebox Management

Systems like OpenNF [32] and its derivatives [33], [34] focus on providing a platform for consistent VNF management, however their focus is on maintaining backward compatibility with currently available VNFs such as Bro [35] for IDS and Squid [36] for caching web proxies. Conversely in this work we set to explore the possibilities opened by a fully programmable networking substrate where also radio access is treated as a standard VNF. Similar considerations can be made also for Split/Merge [37]

III. Network Model

In the VNF placement problem the input consists of Service Function Chains (SFC) consisting of a variable number of VNFs, whereas the substrate network, called Network Function Virtualization Infrastructure (NFVI), provides the physical constraints in terms of bandwidth and capacity [9]. In this context the term capacity is not related only to pure computational resources, such as number of CPU cores, memory, and/or storage, instead it refers also to packet forwarding and radio processing capabilities. Before introducing the proposed solution we need to detail specific notations for the NFVI and the SFC requests.

A. Network Function Virtualization Infrastructure Model

Let $G_{nfvi} = (N_{nfvi}, E_{nfvi})$ be a directed graph modeling the physical network, where N_{nfvi} is the set of $n = |N_{nfvi}|$ physical nodes that compose the substrate network and E_{nfvi}

TABLE I: Substrate network parameters

Variable	Description
G_{nfvi}	Substrate network graph.
N_{nfvi}	Substrate nodes in G_{nfvi}.
E_{nfvi}	Substrate links in G_{nfvi}.
$\omega_c^s(n)$	Available CPU resources at node $n \in N_{nfvi}$.
$\omega_m^s(n)$	Available memory resources at node $n \in N_{nfvi}$.
$\omega_s^s(n)$	Available storage resources at node $n \in N_{nfvi}$.
$\omega_r^s(n)$	Available radio resources at node $n \in N_{nfvi}$.
$\omega_e^s(e^{nm})$	Available resources (e.g. bandwidth) of link $e^{nm} \in E_{nfvi}$.
$\Lambda_n^{c,m,s,r}$	Cost for each unit of node resources, $n \in N_{nfvi}$.
$\Lambda_{e^{nm}}$	Cost for each unit of link resources, $e^{nm} \in E_{nfvi}$.

is the set of edges or links. An edge $e^{nm} \in E_{nfvi}$ if and only if a point–to–point connection exists between $n \in N_{nfvi}$ and $m \in N_{nfvi}$. With respect to the physical network, links are actual wiring media, e.g., an Ethernet cable interconnecting the two nodes[2]. Four weights, $\omega_c^s(n)$, $\omega_m^s(n)$, $\omega_s^s(n)$, $\omega_r^s(n)$, are assigned to each node $n \in N_{nfvi} : \omega_{c,m,s}^s(n) \in \mathbb{N}^+$ and $\omega_r^s(n) \in \mathbb{R}^+, 0 \leq \omega_r^s(n) \leq 1$ representing the packet and radio processing resources *available* on that node. Nodes with all weights equal to 0 (zero) are assumed to be pure packet forwarding nodes. Nodes with $\omega_c^s > 0, \omega_m^s > 0, \omega_s^s > 0$, and $\omega_r^s = 0$ are assumed to be pure packet processing nodes. Finally, nodes with $\omega_c^s = \omega_m^s = \omega_s^s = 0$ and $\omega_r^s > 0$ are assumed to be pure radio access nodes.

Another weight $\omega_e^s(e^{nm})$ assigned to each link $e^{nm} \in E_{nfvi} : \omega_e^s(e^{nm}) \in \mathbb{N}^+$ represents the capacity of the link connecting two nodes. In order to avoid exceeding the nominal capacity of the substrate links, traffic shaping is implemented at the nodes with packet and/or radio processing capabilities. Finally, let P_{nfvi} be the set of all substrate paths and $P_{nfvi}(s,t)$ the shortest path between nodes $s, t \in N_{nfvi}$. Table I summarizes the NFVI parameters.

The weights $\omega_{c,m,s}^s(n)$ associated with the packet processing nodes represent, respectively, the amount of CPU, memory, and storage resources available on that node, while the weights $\omega_r^s(n)$ are specific to the radio access nodes and represent the normalized amount of wireless resources available at that node. Notice how with the term radio access nodes we refer to the generic nodes providing end–users terminals with wireless connectivity and how this model makes no assumptions on the type of resources that can be available at radio node. For example, in a 802.11–based network $\omega_r^s(n)$ could model the amount of airtime available at a certain Access Point (AP). Similarly, in an OFDMA–based network (e.g. LTE), $\omega_r^s(n)$ could be used to model the available radio resources in time *and* frequency at a certain eNodeB (eNB). For the sake of simplicity and without any loss of generality in this work we assume that all radio–enabled nodes initially have the same amount of resources $\omega_r^s(n) = 1, \forall n \in N_{nfvi}$.

A sample substrate network is sketched in Fig. 1. The network is composed by 8 nodes interconnected together. In order to improve readability link weights have been omitted. The substrate network in this example consists of 4 radio access nodes (at the bottom of the picture), and 4 switches, 2 of which supporting just basic forwarding capabilities.

[2]In this work we consider undirected links for simplicity.

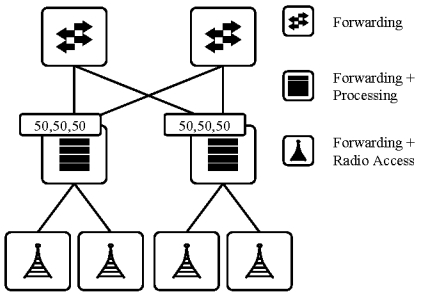


Fig. 1: NFVI network model. The figure shows the three basic virtual resources: forwarding, packet processing, and radio access.

B. Service Function Chain Requests

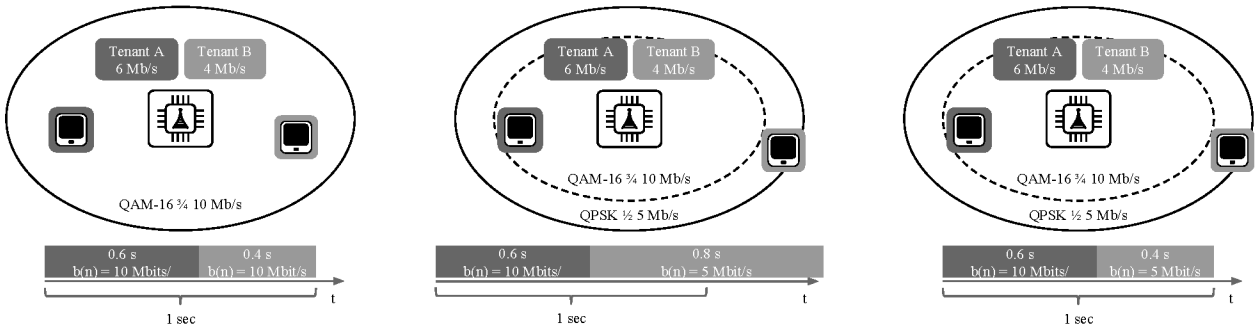
Users are allowed to request SFCs as a *directed* and *acyclic* graphs $G_{sfc} = (N_{sfc}, E_{sfc})$. Where N_{sfc} denotes the set of nodes (i.e. VNFs) and $E_{sfc} \subseteq N_{sfc} \times N_{sfc}$ denotes the set of virtual links. An edge $e^{nm} \in E_{sfc}$ if and only if the packets from VNF $n \in N_{sfc}$ must be forwarded to the VNF $m \in N_{sfv}$. Notice that as opposed to the previous NFVI model, nodes in SFC requests represent virtual network functions through which packets must undergo before leaving the network. Packet processing nodes and links in the SFC request shares the same weights as for the NFVI substrate network ($\omega_c^v, \omega_m^v, \omega_s^v, \omega_e^v$). On the other hand, provisioning of radio resources can be made either in terms of fraction of available radio resources (ω_r^v) or in terms of bandwidth (ω_b^v). In the former case the users will specify the percentage of radio resources they want to be assigned to a certain node, while in the latter case the users will specify the amount of bandwidth assigned to the node. A single SFC request can mix bandwidth–based and resource–based provisioning models.

Due to their stochastic nature, available bandwidth is a time–varying quantity in wireless networks. Channel fading, but also to the distribution of end–users, can greatly influence the network performance. For example, users at the center of the cell can, in general, use more efficient modulations and coding schemes thus achieving higher throughput for a fixed amount of radio resources than users at the edges of the cell. As a result, when the bandwidth–based provisioning model is employed, also the actual channel conditions experienced by the end–users must be taken into account.

Let us call $b(n)$ the actual aggregated throughput of the virtual radio node $n \in N_{sfc}$ in the fraction of resources currently assigned the node. We can then introduce an additional parameter named reference throughput $\Omega_b^v(n) \geq \omega_b^v(n)$ upon which the bandwidth reservation $\omega_b^v(n)$ is enforced. We can then define the effective target bandwidth $\tilde{\omega}_b^v(n)$ for the virtual radio node n as follows:

$$\tilde{\omega}_b^v(n) = \begin{cases} \omega_b^v(n) & \text{if } b(n) \geq \Omega_b^v(n) \\ \omega_b^v(n)\frac{b(n)}{\Omega_b^v(n)} & \text{if } b(n) < \Omega_b^v(n) \end{cases} \quad (1)$$

The parameter $\Omega_b^v(n)$ represents a threshold. When the actual throughput $b(n)$ of the virtual radio node n is above the threshold, then the bandwidth reservation is respected.



(a) Good channel conditions. (b) Poor channel conditions, no isolation. (c) Poor channel conditions, with isolation.

Fig. 2: Impact of channel conditions on slice isolation. Bandwidth demands can be met when channel conditions are good for both slices (a). Conversely, bandwidth targets can not be achieved when one of the tenants is experiencing poor channel conditions (b). Isolation can be preserved by linearly scaling down the bandwidth target for the tenant with adverse channel conditions (c).

Conversely, when $b(n) < \Omega_b^v(n)$ the requested bandwidth $\omega_b^v(n)$ is linearly scaled down. Choosing a small value for $\Omega_b^v(n)$ means that the network can utilize more resource in order to satisfy the bandwidth requirement, which in time could result in an higher pricing. Conversely, a high value for $\Omega_b^v(n)$ means that the network will try to satisfy the bandwidth requirements only for the tenants that are making a better use of the wireless spectrum. Notice how, as it will be more clear in Sec. VI, tenants can make a poor use of the wireless spectrum also due to custom scheduling discipline that favor fairness at the expense of aggregated throughput. This could be a reasonable trade–off in network slices aimed at emergency response where it is critical that all mobile client get the same share of resources even if this could lead to a sub–optimal spectrum utilization.

If we define $\omega_r^v(n) = \frac{\tilde{\omega}_b^v(n)}{b(n)}$, then, in order for the SFC request to be feasible, it must hold:

$$\sum_{n \in N_{sfc}^b} \frac{\tilde{\omega}_b^v(n)}{b(n)} + \sum_{n \in N_{sfc}^r} \omega_r^v(n) \leq \omega_r^v(m) = 1 \qquad (2)$$

Which means that the sum of the fractions of radio resources allocated to virtual nodes must be less than or equal to the resources available at the substrate node m. Table II summarizes the SFC request parameters.

For example, consider the case depicted in Fig. 2a. Here two tenants ($A\&B$) requested two slices with aggregated capacity of, respectively, 6 Mb/s and 4 Mb/s, while the reference bandwidth (Ω_b^v) for both tenants has been set to 10 Mb/s. Moreover, for simplicity and without losing generality, let us assume that only one client is active in either slice. If the cell capacity $b(n)$ experienced by both clients is equal to, or higher than, 10 Mb/s it is easy to see how both slices can be accommodated by the system, thus in this case $b(n) \geq \Omega_b^v(n)$ and $\tilde{\omega}_b^v(n) = \omega_b^v(n)$ for both slices.

However, if the cell capacity experienced by the wireless client in the tenant B's slice is reduced to 5 Mb/s, then the same request can not be accommodated anymore. This is due to the fact that serving the same amount of data to the wireless client in the tenant B's slice now requires twice the amount of radio resources it did before (see Fig 2b). In this case since $b(B) < \Omega_b^v(B)$, thus:

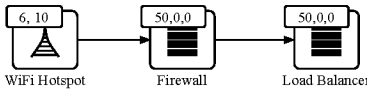
$$\tilde{\omega}_b^v(B) = \omega_b^v(B)\frac{b(B)}{\Omega_b^v(B)} = 4\frac{5}{10} = 2$$

This results in the resource allocation sketched in Fig 2c which penalizes only the performance of the tenant B slice without affecting tenant A's slice. It is worth noticing that this example can be easily generalized to an OFDMA system were resources are assigned across a time/frequency matrix.

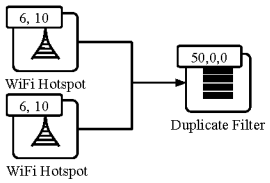
A few sample SFC requests are sketched in Fig. 3. Notice that wireless terminals are not represented in that they are outside the control of the orchestration framework. The linear SFC request in Fig. 3a consists of three VNFs including a WiFi hotspot, a firewall, and a load balancer. The WiFi hotspot request will also include parameters such as the name of the network and the authentication parameters (e.g. type of encryption, RADIUS server to be used, etc.) however these kind of information are purely functional and are thus omitted in this section. The SFC request in Fig. 3b implements a performance enhancing network service, namely duplicates filtering in dense urban scenarios. Finally, the SFC request in Fig. 3c accounts for an access enforcement scenario whereby multiple security related VNFs are deployed in parallel. Notice how all radio access VNF use the bandwidth–based resource provisioning model.

TABLE II: Service function chain request parameters

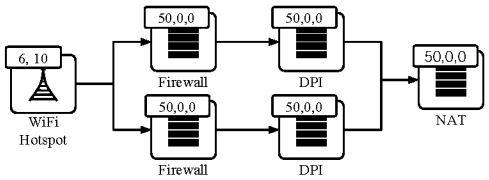
Variable	Description
G_{sfc}	Service function chain graph.
N_{sfc}	Virtual nodes in G_{sfc}.
E_{sfc}	Virtual links in G_{sfc}.
$\omega_c^v(n)$	Requested CPU resources at node $n \in N_{sfc}$.
$\omega_m^v(n)$	Requested Memory resources at node $n \in N_{sfc}$.
$\omega_s^v(n)$	Requested Storage resources at node $n \in N_{sfc}$.
$\omega_r^v(n)$	Requested Radio resources at node $n \in N_{sfc}$.
$\omega_b^v(n)$	Requested Bandwidth at node $n \in N_{sfc}$.
$\Omega_b^v(n)$	Reference bandwidth at node $n \in N_{sfc}$.
$\omega_e^v(e^{nm})$	Requested resources (e.g. bandwidth) of link $e^{nm} \in E_{sfc}$.



(a) Radio access network sharing (Linear topology).



(b) Duplicate filtering (Branched topology).



(c) Load–balancing (Loop topology).

Fig. 3: Sample SFC Requests. Notice how resource requests for all radio VNFs are bandwidth–based.

C. Virtual Network Function Placement

In this section we shall provide the optimal ILP formulation for the SFC embedding problem, while in the next section we will present a scalable heuristic. The overall objective is to compute the optimal VNF placement based on the available radio resources. The chosen objective function is:

$$min\Big(\sum_{n\in N_{nfvi}}\sum_{n'\in N_{sfc}}\Big(w_c^v(n')\Lambda_n^c + w_m^v(n')\Lambda_n^m + w_s^v(n')\Lambda_n^s + w_r^v(n')\Lambda_n^r\Big)\Phi_n^{n'} + \sum_{e\in E_{nfvi}}\sum_{e'\in E_{sfc}}\omega_e^v(e')\Lambda_e\Phi_e^{e'}\Big)$$

where, $\Phi_n^{n'}$, $\Phi_e^{e'} \in 0,1$ are two binary variables indicating respectively if the VNF $n' \in N_{sfv}$ has been mapped to node $n \in N_{nfvi}$ and if the virtual link $e' \in E_{sfc}$ has been mapped to the substrate link $e \in E_{nfvi}$.

A valid solution is the one where the resources utilized by the SFC request are at most equal to the available resources on the substrate network nodes:

$$\sum_{n'\in N_{sfc}} \omega_c^v(n')\Phi_n^{n'} \leq \omega_c^s(n) \quad \forall n \in N_{nfvi} \tag{3}$$

$$\sum_{n'\in N_{sfc}} \omega_m^v(n')\Phi_n^{n'} \leq \omega_m^s(n) \quad \forall n \in N_{nfvi} \tag{4}$$

$$\sum_{n'\in N_{sfc}} \omega_s^v(n')\Phi_n^{n'} \leq \omega_s^s(n) \quad \forall n \in N_{nfvi} \tag{5}$$

$$\sum_{n'\in N_{sfc}} \omega_r^v(n')\Phi_n^{n'} \leq \omega_r^s(n) \quad \forall n \in N_{nfvi} \tag{6}$$

and links:

$$\sum_{e'\in E_{sfc}} \omega_e^v(e')\Phi_e^{e'} \leq \omega_e^s(e) \quad \forall e \in E_{nfvi} \tag{7}$$

In this work we also assume that every VNF in the SFC request shall be mapped to a different substrate node:

$$\sum_{n'\in N_{sfc}} \Phi_n^{n'} \leq 1 \quad \forall n \in N_{nfvi} \tag{8}$$

Every VNF in the SFC request shall be mapped only once:

$$\sum_{n\in N_{nfvi}} \Phi_n^{n'} = 1 \quad \forall n' \in N_{sfc} \tag{9}$$

In terms of radio resources requirements, the following constrain, deriving from (1) and (2), enforces that for every radio processing node $n \in N_{nfvi}$ a feasible request has been made:

$$\sum_{n'\in N_{sfc}^b} \frac{\omega_b^v(n')}{\Omega_b^v(n')}\Phi_n^{n'} + \sum_{n'\in N_{sfc}^r} \omega_r^v(n')\Phi_n^{n'} \leq 1 \quad \forall n \in N_{nfvi} \tag{10}$$

Finally, the following constraint enforces that for each link $e^{nm} \in E_{sfc}$ there must be a continuous path allocated between the pair of physical nodes on top of which the VNFs $n, m \in N_{sfc}$ have been mapped.

$$\sum_{j\in N_{nfvi}}^{j>i} \Phi_{e^{ij}}^{e^{nm}} - \sum_{j\in N_{nfvi}}^{j<i} \Phi_{e^{ji}}^{e^{nm}} = \Phi_i^n - \Phi_i^m \tag{11}$$

$$\forall i \in N_{nfvi} \quad \forall e^{nm} \in E_{sfc}$$

IV. Heuristic

The ILP formulation described in the previous sections can not be applied to realistic scenarios due to its limited scalability. For example, embedding a 6–nodes linear SFC request over a $k = 8$ fat–tree substrate topology can take several hours on Intel Core i7 laptop (3.0 GHz CPU, 16 Gb RAM) using the Matlab® ILP solver (intlinprog). Notice also that even a $k = 8$ fat–tree substrate topology is rather small if compared with realistic deployments where it is common to find $k = 24$ fat–tree networks, making the ILP problem formulation intractable. In this section we present a heuristic, named *WiNE*, that can handle similar (and more complex) requests in tens of minutes.

The proposed heuristic is composed of three steps. In the first step for each virtual node $n \in N_{sfc}$, the heuristic computes the list of candidate substrate nodes (see Alg. 1). These are the substrate nodes that can support the virtual nodes in the SFC request given the input capacity constraints. In the second step, the virtual nodes $n \in N_{sfc}$ are sorted in decreasing order according to the number of candidate substrate nodes (see Alg. 2). In the third step, the sorted list of virtual nodes is traversed starting with the virtual node with more embedding opportunities. For each candidate substrate node $p \in N_{nfvi}$ the heuristic considers all the neighboring nodes $m \in N_{sfc}$ of the virtual node n. The heuristic then assigns the node n to the substrate node $m \in N_{nfvi}$ with the lowest virtual edge mapping cost W (see Alg. 3).

Algorithm 1 Compute list of candidate substrate nodes

```
procedure FindCandidates(N_nfvi, N_sfc)
    for n ∈ N_sfc do
        for p ∈ N_nfvi do
            if ω^s_{c,m,s,r}(p) >= ω^v_{c,m,s,r}(n) then
                n.candidates.add(p)
            end if
        end for
    end for
end procedure
```

Algorithm 2 Sort list of candidate substrate nodes

```
procedure SortCandidates(N_sfc)
    sort(N_sfc)
end procedure
```

Algorithm 3 Nodes and links assignment

```
procedure NodeAndLinkAssignment(G_nfvi, G_sfc)
    for n ∈ N_sfc do
        for p ∈ n.candidates do
            if p.used then
                continue
            end if
            for m ∈ n.neighbors do
                if m.mapped then
                    cost = W(e^nm, p, m.mapped)
                else
                    cost = +∞
                    for q ∈ m.candidates do
                        cost = min(cost, W(e^nm, p, q))
                    end for
                end if
                mapping_cost(p)+ = cost
            end for
        end for
        p ← argmin(mapping_cost(p))
        n.mapped ← p
        p.used ← True
        for m ∈ n.neighbors do
            if m.mapped then
                Allocate path P_nfvi(n.mapped, m.mapped)
            end if
        end for
    end for
end procedure
```

We define the virtual edge mapping cost $W : E_{sfc} \times N_{nfvi} \times N_{nfvi} \rightarrow \mathbb{R}$ between a virtual edge $e^{nm} \in E_{sfc}$ and a pair of substrate nodes $p, q \in N_{nfvi}$ as follows:

$$W(e^{nm}, p, q) = \sum_{e \in P_{nfvi}(p,q)} \Lambda_{e^{nm}} \omega_e^v(e^{nm})$$

This represents the cost of embedding the virtual edge $e^{nm} \in E_{sfc}$ over the path $P_{nfvi}(p, q)$ between the substrate nodes $p, q \in N_{nfvi}$ given that virtual nodes n, m are mapped on, respectively, the substrate nodes p, q. Minimizing the virtual edge mapping cost essentially means that substrate nodes that are far away from node's m embedding opportunities are penalized. This results in virtual nodes in an SFC request to be placed close to each other over the substrate network, which in time means that more resources can be put offline when the system is scarcely loaded.

V. Evaluation

The goal of this section is to compare the *relative* performance of the ILP–based placement algorithm with the performance of our placement heuristic using different synthetic substrate network and different SFC requests. In this section we shall first describe the simulation environment and then the performance metrics. Simulations are carried out in a discrete event simulator implemented in Matlab®.

A. Simulation Environment

The ILP–based placement algorithm and the proposed placement heuristic are evaluated in three different scenarios. In the first scenario, linear VNF requests, similar to the one depicted in Fig. 3a, are considered. In the second scenario, branched VNF requests, similar to the one depicted in Fig. 3b, are considered. Finally, in the third scenario, VNF requests with loops, similar to the one depicted in Fig. 3c, are used. The number of VNFs in each SFC request as well as the actual amount of radio, computational, memory, storage, and link resources are randomly generated for each request.

The reference substrate network is k–ary fat–tree with $k = 4, 6, 8$, where leaf nodes are WiFi Access Points (APs) rather than servers. This results in a total of, respectively, 16, 54, and 128 WiFi APs. The computational, memory, storage, radio, and link resources for the substrate network are initially all set to 100. The cost of using each unit of node $\Lambda_n^{c,m,s,r}$ and link Λ_e resources is set to 1.

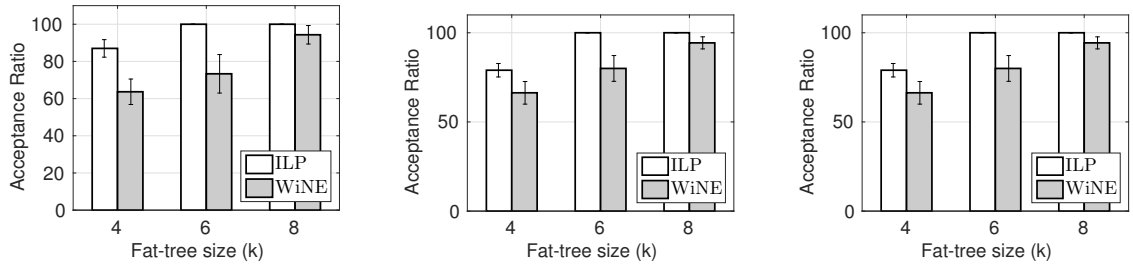
The number of VNFs in each SFC request depends on the SFC type. In the case of linear and branched SFC requests the number is randomly picked in the set $\{3, 6\}$, while in the case of cyclic SFC requests the number is randomly picked in the set $\{4, 6\}$. The computational, memory, and storage requirements for each SFC requests are uniformly distributed between $[5, 30]$, while the radio and link requirements are uniformly distributed between $[5, 60]$.

The metrics used in this study are the standard ones adopted in several other related works (see, e.g., [13], [38], [39]). For each scenario the number of accepted requests, the average embedding cost, the average node and link utilization, and the execution time using either the ILP–based placement or the proposed heuristic are considered.

In this study we assume that a fixed number of SFC requests are embedded sequentially onto the substrate network. In particular in each run, the simulator tries to embed 30 randomly generated SFC requests. Reported results are the average of 10 simulations.

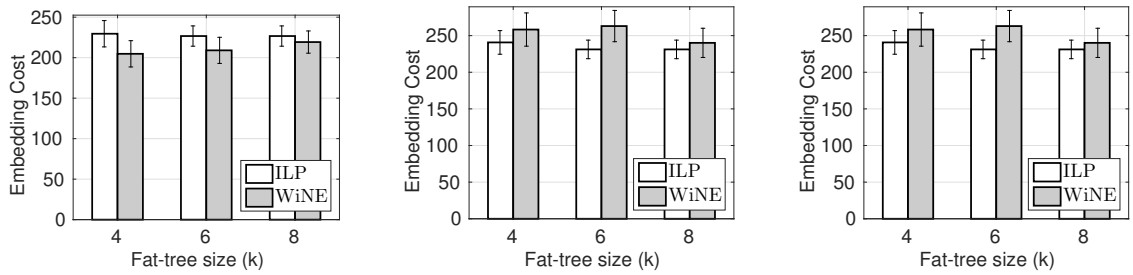
B. Simulation Results

Figures 4 and 5 shows the percentage of accepted SFC requests for different substrate networks and the average embedding cost. As expected the ILP–based placement algorithm is more efficient than *WiNE* in mapping the incoming requests. This can be seen in terms of both a higher number of accepted requests as well as a lower average embedding cost. Notice however that, at least for linear SFCs, *WiNE* actually has a lower embedding cost. This does not mean that *WiNE* is more efficient than the ILP–based algorithm but rather that, by



(a) Linear. (b) Branched. (c) Loop.

Fig. 4: Acceptance ratio using the ILP–based algorithm and the heuristics with different virtual and substrate topologies.



(a) Linear. (b) Branched. (c) Loop.

Fig. 5: Average embedding cost using the ILP–based algorithm and the heuristics with different virtual and substrate topologies.

accepting a lower number of SFC requests, *WiNE* also utilizes, on average, less substrate resources. It is also worth noticing that the efficiency of the proposed heuristic increases with the size of the substrate network. This hints toward the fact that *WiNE* may be capable of closing the gap with the ILP–based placement algorithm for realistic substrate networks.

Figure 6, 7 and 8 summarize the substrate resource utilization. As it can be noticed the ILP–based placement algorithm is characterized by a higher utilization ratio for both radio and computational nodes. This results in fewer substrate nodes being used to support the same number of SFC requests which in time could result in a more energy efficient operation if unused nodes are turned–off. Notice that this is further supported by Fig. 8 where the substrate links utilization is reported. As it can be seen, *WiNE* is characterized by an higher link utilization, which means that the proposed heuristic is less efficient in finding shorter paths between VNF. However, it is also worth noticing that the gap between the ILP–based placement algorithm and *WiNE* gets smaller as the size of the substrate network increase.

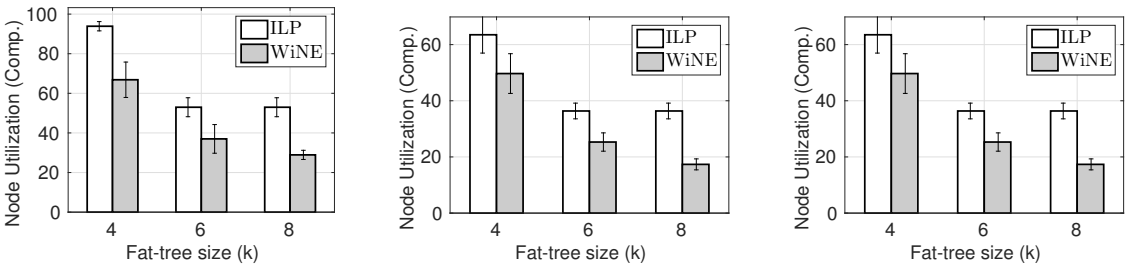
Figure 9 shows that the average amount of time required to embed a *single* SFC request using the ILP–based placement algorithm is significantly higher than the time required to embed the same request using *WiNE*. The ILP problem becomes essentially intractable for substrate networks with more than a few tens of nodes (irrespective of the number of VNFs in the request), while *WiNE* can effectively embed complex SFC requests on substrate networks with hundreds of nodes in a limited amount of time.

VI. Implementation

A. Overview

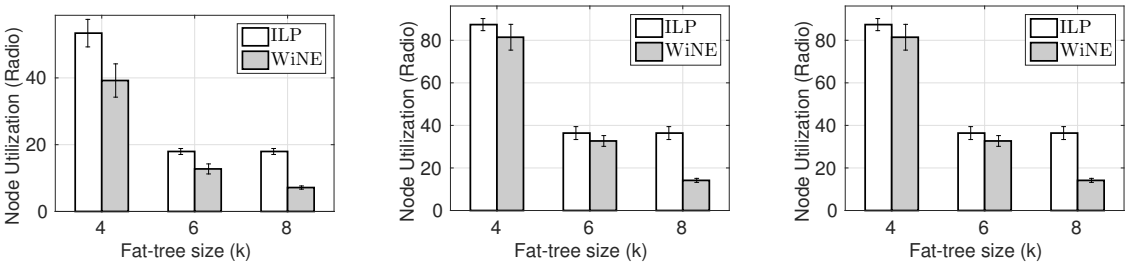
We implemented the VNF placement and scheduling solution presented in this work in a proof–of–concept NFV management and orchestration framework, named *EmPOWER*. Notice that the prototype currently targets only wireless access networks based on the 802.11 family of standards and, as a consequence, the applications described in the next section target Enterprise WLAN and Campus network scenarios. Nevertheless, as seen in the previous sections, the provisioning model does not make any assumption about the particular link–layer technology and can be as well applied to any kind of radio access network including OFDMA networks such as LTE and LTE–Advanced.

Our proof–of–concept is loosely modeled after the ETSI reference NFV Architecture [9]. As it can be seen in Fig. 10, the architecture is conceptually divided into three layers. The bottom layer consists of the physical as well as the virtualized resources composing the NFVI. In the second layer we have the actual VNFs which are the software implementation of a particular network function capable of being executed over the NFVI. We remind the reader that in this work also radio access is treated as a VNF. Finally, in the third layer we have the Operational Support System (OSS) and the Business Support System (BSS) used by the network administrators to operate and manage their virtual networks. The Management and Orchestration plane covers the orchestration and the management of physical and/or virtual resources that support the NFVI as well as the life–cycle management of the VNFs, i.e. creation, configuration, monitoring, and destruction.



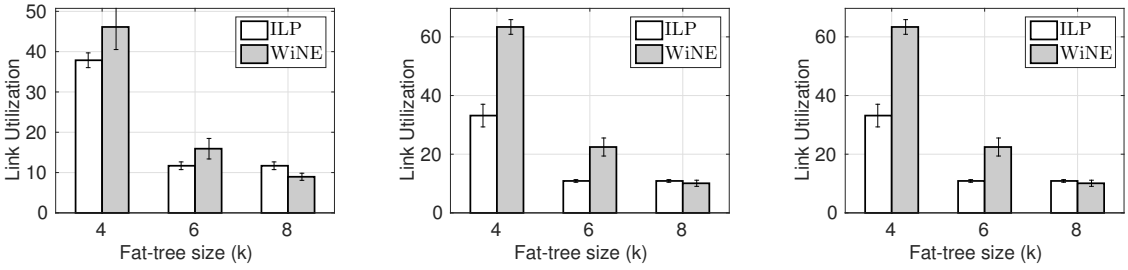
(a) Linear. (b) Branched. (c) Loop.

Fig. 6: Average computational nodes utilization using the ILP–based algorithm and the heuristics with different virtual and substrate topologies.



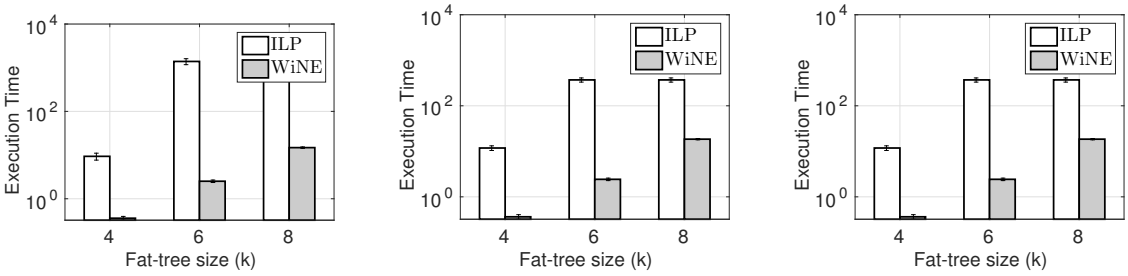
(a) Linear. (b) Branched. (c) Loop.

Fig. 7: Average radio nodes utilization using the ILP–based algorithm and the heuristics with different virtual and substrate topologies.



(a) Linear. (b) Branched. (c) Loop.

Fig. 8: Average link utilization using the ILP–based algorithm and the heuristics with different virtual and substrate topologies.



(a) Linear. (b) Branched. (c) Loop.

Fig. 9: Average execution time using the ILP–based algorithm and the heuristic with different virtual and substrate topologies.

B. Network Function Virtualization Infrastructure

Our architecture currently accounts for three kinds of NFVI resources, namely: basic forwarding nodes (i.e. OpenFlow switches), packet processing nodes, and radio access nodes. The latter, in addition to the features supported by the packet processing node, also embed specialized hardware in the form of one or more 802.11 Wireless NICs. Figure 11 sketches the system architecture.

We name Wireless Termination Points (WTPs) the physical points of attachment in the RAN (e.g. WiFi Access Points or LTE eNodeBs) supporting virtualized radio processing capabilities. Conversely, the Click Packet Processors (CPPs) are the forwarding nodes with computational capacity. These nodes are essentially programmable switches running an embedded version of Linux and capable of performing arbitrary operations on the traffic, e.g. load–balancing, firewalling, deep

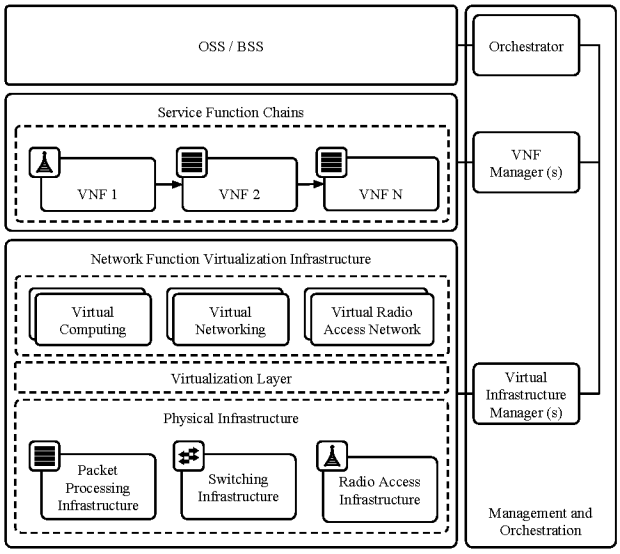


Fig. 10: Reference network function virtualization architecture [9].

packet inspection, etc.

As the name suggests the actual packet processing is performing by multiple instances of the Click Modular Router [40]. Click allows to build complex VNFs using simple and reusable components, called *elements*. Click includes over 300 elements supporting functions such as packet classification, access control, deep packet inspection. Elements can be composed in order to realize complex VNFs. Finally, Click is easily extensible with custom processing elements making it possible to support features that are not provided by the standard elements.

Each WTP/CPP includes an OpenVSwitch instance, one or more VNFs, and one Agent. The latter is in charge of monitoring the status of each VNF as well as handling requests coming from the controller. In the current implementation the monitoring features includes: number of packets/bytes transmitted and received as well as the amount of resources utilized (cpu time, memory, storage) by each VNF.

Finally, in the current prototype CPPs are built upon the Soekris 6501-70 platform (1.6 GHz Intel Atom CPU, 2 Gbyte of RAM) equipped with 12 Gigabit Ethernet Ports. On the other hand WTPs exploit the ALIX 2D platform (500 MHz AMD Geode CPU, 256 Mb of RAM) equipped with two 802.11/a/b/g/n Wireless NICs. Both platforms run an embedded Linux distribution, namely OpenWRT Barrier Breaker.

C. Virtual Infrastructure Managers

As Virtual Infrastructure Managers (VIMs) we use a combination of frameworks. Ryu [41] is used to configure the switching fabric while for the *Programmable Network Fabric* we extended the EmPOWER SD–RAN controller presented by the authors in [42] in order to support also the generalized packet processing nodes. The controller supports multiple slices on top of the same physical infrastructure. A slice is a virtual network with its own set of *WTPs*/*CPPs*. The controller supports multiple south–bound protocols, namely the *LVAP* Protocol and the *LVNF* Protocol for communicating with, respectively, *WTPs* and *CPPs*. The controller is responsible for the deploying the VNFs on the network devices.

D. Orchestrator

From an architectural standpoint, the VNF placement algorithm resides in the Orchestrator which is in charge of deciding whether a particular request can be accepted or if it must be refused. If a request is accepted, then the Orchestrator is in charge of mapping the request onto the substrate network, i.e., network resources must be allocated and configured on both the substrate nodes and the substrate links and the VNFs must be instantiated on the selected nodes (see Fig. 11). Notice how, our proof–of–concept lacks a proper Orchestrator and that the VNF placement is computed offline using Matlab®.

VII. Applications

In this section we shall describe three SFCs implemented and tested over a small scale testbed deployed at CREATE–NET premises. The testbed consists of 2 OpenFlow–switches, 2 *CPPs*, and 20 *WTPs*.

A. Data/Management Plane Offloading

Wireless, and in particular mobile networks, have been so far designed around the requirements of the downlink (i.e. cell or access point selection is performed using downlink signal strength measurements). In the recent years, however, we have witnessed a mushrooming of new uplink–centric applications such as Machine Type Communications (MTC), Internet of Things (IoT), and Vehicle to Infrastructure (V2I) as well as of symmetric mobile applications. This calls for a paradigm shift where the traffic originated from a wireless client is received by one node while the traffic destined to the same client is transmitted by another node. This kind of network setup is usually referred to as uplink/downlink decoupling and, in its most general form, can consists of two possibly non overlapping sets of transmitting and receiving nodes.

The previously introduced *EmPOWER* platform [42] allows for mobility management policies whereby a wireless client downlink can be scheduled on one WiFi AP while the uplink can be scheduled on *one or more* WiFi APs. This feature allows to exploit the broadcast nature of the wireless medium and to opportunistically receive the same transmission at multiple in–range APs. However, if not properly controlled such a feature can lead to an overload in the network core. For example, a wireless client scheduled on N APs in the uplink direction could increase the load on the network core by a factor of N. Moreover, a straightforward implementation of such a mechanism could generate a significant increase in the number of duplicate packets which in time could trigger at the transport layer.

In order to address this issue, we implemented a VNF which filters out duplicate 802.11 frames based on their sequence number. Notice how, this is a particular example of MAC/LLC offloading. Figure 12 sketches a slightly more generic case were both data–plane (duplicates filtering) and management (probe response generation) are offloaded to two different

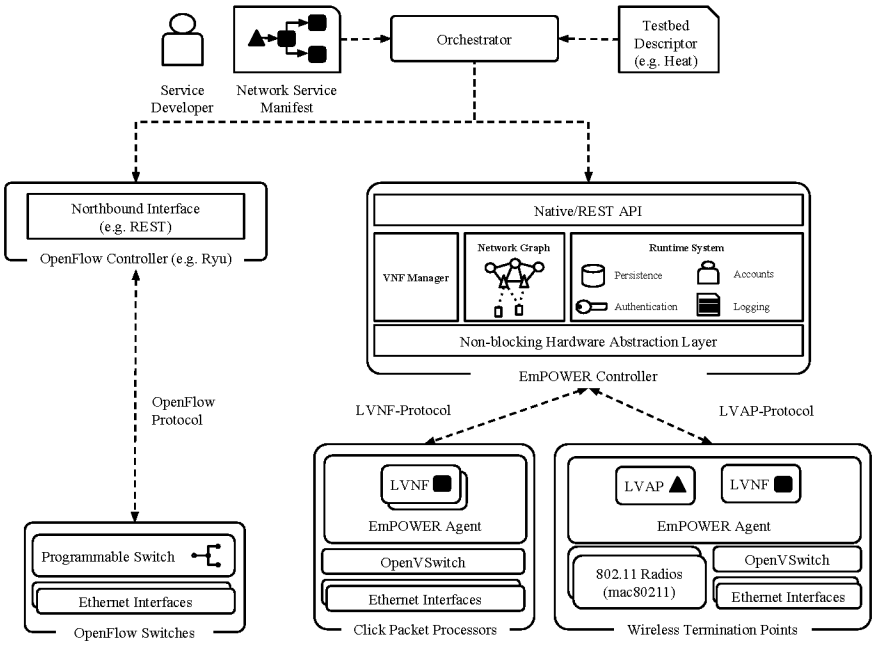


Fig. 11: The *EmPOWER* network architecture.

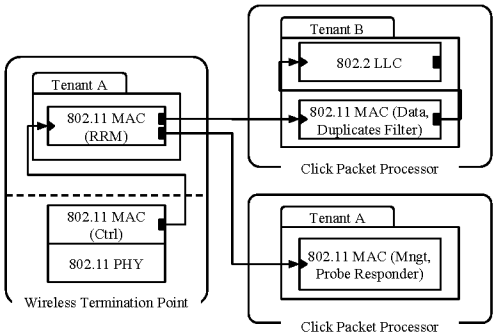


Fig. 12: Data/Management Plane Offloading application. Data–plane (duplicates filtering) and Management–plane (probe response generation) operations are offloaded to two different *CPP*s.

*CPP*s. In this section however we shall evaluate only the performances of the duplicate filtering VNF.

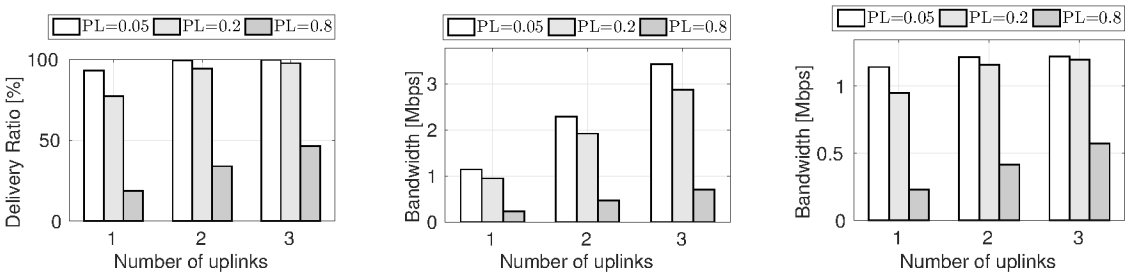
Traffic originated from clients is received at one or more *WTP*s where it is encapsulated (802.11 over Ethernet) and then forwarded to a *CPP* where the duplicates filtering VNF is deployed. This VNF is also responsible for decapsulating the 802.11 frame and converting it in an Ethernet frame before forwarding it to its intended destination.

In order to evaluate this VNF we exploited a network setup composed of a single client and three APs. Traffic is injected from the wireless client as a single UDP stream. Packet transmission rate and payload are kept fixed at, respectively, 100 packets/s and 1472 bytes. Impairments on the link between client and APs are simulated by randomly dropping received frames with probability p at all receiving APs. For all measurements a total of 6000 frames were generated. Confidence intervals were very small for all the data points and have therefore be omitted in order to improve readability. Measurements have been taken using three packet dropping probability, namely: 0.05 (good channel conditions), 0.2 (medium channel conditions), and 0.8 (poor channel conditions)

As it can be seen from Fig. 13a, the end–to–end packet delivery ratio increases with the number of available uplinks. The proposed uplink/downlink decoupling solution can provide a small performance improvement even when the channel conditions are good ($PL = 0.05$). On the other hand the performance improvements are significant when the channel condition get worse. In particular this solution allows to turn an essentially broken channel (4 out of 5 dropped packet) into an usable channel (1 out of 2 dropped packets). Finally, in Fig. 13b and Fig. 13c we can see the impact of the duplicate filtering VNF on the bandwidth utilization. As expected without the VNF the bandwidth utilization increases with the number of uplink, while using the VNF filtering the bandwidth utilization does not exceed the nominal goodput.

B. Radio Access Network Slicing

In this use case we aim at demonstrating the performance isolation features enabled by our joint VNF placement and scheduling solution. Such use case is by introducing a MAC Hypervisor implementing the slice scheduling techniques described in the previous sections. Figure 14) sketches the internal architecture of a radio access node. Notice how, not the entire WiFi MAC is virtualized. More precisely, WiFi control frames generation (e.g. ACK and RTS/CTS) as well as the actual physical layer are not virtualized due to their tight timing/computational constraints. On the other hand each tenant can have its own Radio Resource Management (RRM) instance as well as its own management place. In this work with RRM we refer to wireless client scheduling and to rate



(a) Packet Loss. (b) BW w/o VNF. (c) BW w/ VNF.

Fig. 13: Packet loss (a) and network utilization (b and c) for a single client scheduled on multiple APs on the uplink direction.

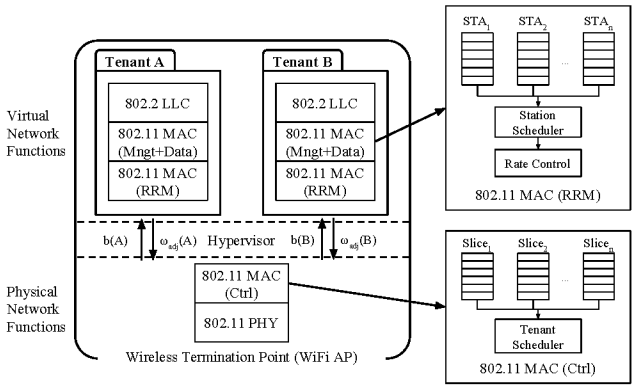
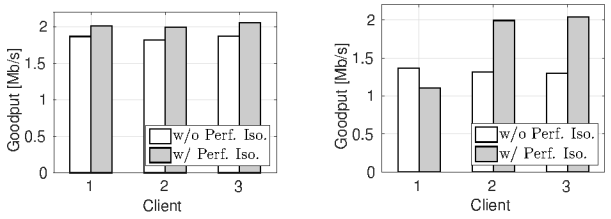


Fig. 14: Radio Access Network Slicing Application.



(a) Good channel. (b) Poor channel.

Fig. 15: Isolation across clients in different networks. Client 1 and 2 are in Tenant 1's network, while Client 3 is in Tenant 2's network.

adaptation, which can be defined on a per–tenant basis[3]. For example, one tenant can use a wireless station scheduler aimed at ensuring fairness while another tenant could opt for a scheduler aimed at improving the aggregated slice bandwidth.

The hypervisor monitors the channel utilization as well as the amount of traffic generated by each Tenant. Such information is use to compute the effective aggregated goodput of each slice $b(n)$. It is worth stressing both channel activity and goodput are two quantities that can be monitored by the hypervisor with low overhead and without affecting the operation of the RRM policy implemented by each Tenant. For example a Tenant with one wireless user experiencing poor channel conditions will see its goodput reduced due to re–transmissions. Similarly a Tenant implementing wireless clients scheduling policies aimed at maximizing fairness will also see a reduction in goodput if some clients use in either the downlink or the uplink direction less efficient modulation and coding schemes.

The network setup used for the measurements consists of two tenants $A\&B$ each requesting a single hotspot. The bandwidth request coming from Tenant A & B is, respectively 4 and 2 Mb/s, while the reference bandwidth (Ω_b^v) is 6 Mb/s for both tenants. A total of three wireless clients is active in the networks. Clients 1 and 2 belongs to Tenant A while Client 3 belongs to Tenant B. Traffic is generated from a server sharing the same backhaul with the AP and consists of three UDP stream (one for each client). Each stream has constant inter–departure time and packet size resulting in a transmission rate of 10 Mb/s for each stream. In order to simulate an hotspot with limited capacity the rate control algorithm used by the access point has been modified in order to always use the lowest transmission rate (6 Mb/s). Measurements have been carried in two different scenarios differentiated by the channel condition experienced by client number 1 which is positioned in such a way to experience channel condition raging from *Good* to *Poor*.

As it can be seen from Fig. 15a, when Client 1 is experiencing good channel conditions the proposed VNF scheduling method can satisfy the bandwidth request for both tenants (notice that Tenant A request of 4 Mb/s has been equality partitioned among the two clients). On the other hand, when Client 1 starts experiencing poor channel conditions (see Fig. 15b) the legacy resource provisioning mechanism allocates the same bandwidth to all the clients. This behavior, known as IEEE 802.11 performance anomaly [43], allows a node which experiences poor channel conditions to monopolize the wireless medium lowering the performance of the whole system. Conversely, the proposed resource provisioning mechanism can meet the bandwidth reservations made by the two tenants by linearly scaling down the amount of resources allocated to the tenant that is experiencing poor channel conditions.

VIII. Conclusions

NFV is rapidly emerging as a flexible solution to deploy and operate large IT infrastructure. However, in order to fully deliver on its promises, the concept of NFV must be extended also to the radio access segment of the mobile network. In this paper we tackled this challenge by presenting a novel formulation of the VNF placement problem encompassing

[3]Notice how, the hypervisor implementation details are out of the scope of this work and are thus omitted.

also radio VNFs. We introduced then a ILP–based algorithm for small networks and a scalable heuristic, named *WiNE*, for larger deployments. Finally, we reported on a preliminary proof–of–concept implementation of a NFV Management and Orchestration framework for Enterprise WLANs.

As future work we plan to investigate the resiliency properties of *WiNE* in case of nodes and link failures and to study how VNF placement can be optimized by taking into account wireless clients distribution. We also plan to verify the applicability of our problem formulation to OFDMA–based radio access networks like LTE and LTE–Advanced and to extend the prototype (both the hypervisor and the controller) adding support for VNF migration and scaling as well as for additional virtualized substrate resources including cellular technologies and Mobile Edge Computing capabilities.

ACKNOWLEDGMENT

This paper has received funding from the European Union's Horizon 2020 research and innovation programme under 5G-PPP COHERENT project (Grant Agreement No. 671639).

REFERENCES

[1] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. of IEEE CNSM*, 2014.

[2] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[3] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, pp. 1–53, 2014.

[4] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic Virtual Network Function Placement," in *Proc. of IEEE CloudNet*, Niagara Falls, Canada, 2014.

[5] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, and T. Ahmed, "Virtual Network Functions Orchestration in Wireless Networks," in *Proc. of IEEE CNSM*, Barcelona, Spain, 2015.

[6] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "Routebricks: Exploiting parallelism to scale software routers," in *Proc. of ACM SOSP*, 2009.

[7] S. Han, K. Jang, K. Park, and S. Moon, "Packetshader: A gpu-accelerated software router," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 195–206, Aug. 2010.

[8] K. K. Ram, A. L. Cox, M. Chadha, and S. Rixner, "Hyper-switch: A scalable software virtual switching architecture," in *Proc. of USENIX ATC*, 2013.

[9] E. T. S. I. (ETSI), *ETSI GS NFV 002 Network Functions Virtualisation (NFV); Architectural Framework*, December 2014.

[10] "OPNFV: Open Platform for Network Function Virtualization." [Online]. Available: https://www.opnfv.org/

[11] "OpenMANO." [Online]. Available: https://github.com/nfvlabs/openmano

[12] "OpenBATON." [Online]. Available: http://openbaton.github.io/

[13] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 1, pp. 206 –219, February 2012.

[14] M. Chowdhury, F. Samuel, and R. Boutaba, "Polyvine: policy-based virtual network embedding across multiple domains," in *Proc. of ACM VISA*, 2010.

[15] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual network embedding with opportunistic resource sharing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 3, pp. 816–827, March 2014.

[16] M. A. Khan, N. Shahriar, R. Ahmed, and R. Boutaba, "Simple: Survivability in multi-path link embedding," in *Proc. of IEEE CNSM*, Barcelona, Spain, 2015.

[17] D. Breitgand, A. Epstein, A. Glikson, A. Israel, and D. Raz, "Network aware virtual machine and image placement in a cloud," in *Proc. of IEEE CNSM*, 2013.

[18] M. Barshan, H. Moens, and F. De Turck, "Design and evaluation of a scalable hierarchical application component placement algorithm for cloud resource allocation," in *Proc. of IEEE CNSM*, 2014.

[19] M. Barshan, H. Moens, S. Latre, and F. De Turck, "Algorithms for efficient data management of component-based applications in cloud environments," in *Proc. of IEEE NOMS*, 2014.

[20] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, "A novel approach to virtual networks embedding for sdn management and orchestration," in *Proc. of IEEE NOMS*, 2014.

[21] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *Proc. of IEEE NOMS*, 2014.

[22] F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Proc. of IEEE CNSM*, Barcelona, Spain, 2015.

[23] L. Xia, S. Kumar, X. Yang, P. Gopalakrishnan, Y. Liu, S. Schoenberg, and X. Guo, "Virtual wifi: Bring virtualization from wired to wireless," *SIGPLAN Not.*, vol. 46, no. 7, pp. 181–192, Mar. 2011.

[24] P. Lv, X. Wang, and M. Xu, "Virtual access network embedding in wireless mesh networks," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1362–1378, Sep. 2012.

[25] P. Lv, Z. Cai, J. Xu, and M. Xu, "Multicast service-oriented virtual network embedding in wireless mesh networks," *Communications Letters, IEEE*, vol. 16, no. 3, pp. 375–377, March 2012.

[26] G. Bhanage, R. Daya, I. Seskar, and D. Raychaudhuri, "Vnts: A virtual network traffic shaper for air time fairness in 802.16e systems," in *Proc. of IEEE ICC*, May 2010.

[27] G. Bhanage, D. Vete, I. Seskar, and D. Raychaudhuri, "Splitap: Leveraging wireless network virtualization for flexible sharing of wlans," in *Proc. of IEEE GLOBECOM*, Dec 2010.

[28] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "Nvs: A substrate for virtualizing wireless resources in cellular networks," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 5, pp. 1333–1346, Oct 2012.

[29] ——, "Cellslice: Cellular wireless resource slicing for active ran sharing," in *Proc. of IEEE COMSNETS*, 2013.

[30] G. Bhanage, I. Seskar, R. Mahindra, and D. Raychaudhuri, "Virtual basestation: Architecture for an open shared wimax framework," in *Proc. of ACM VISA*, 2010.

[31] G. Smith, A. Chaturvedi, A. Mishra, and S. Banerjee, "Wireless virtualization on commodity 802.11 hardware," in *Proc. ACM WinTECH*, 2007.

[32] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "Opennf: Enabling innovation in network function control," in *Proc. of ACM SIGCOMM*, Chicago, Illinois, USA, 2014.

[33] B. Kothandaraman, M. Du, and P. Sköldström, "Centrally controlled distributed vnf state management," in *Proc. of ACM HotMiddlebox*, London, United Kingdom, 2015.

[34] A. Gember-Jacobson and A. Akella, "Improving the safety, scalability, and efficiency of network function state transfers," in *Proc. of ACM HotMiddlebox*, London, United Kingdom, 2015.

[35] "The Bro Network Security Monitor." [Online]. Available: https://www.bro.org/

[36] "The Squid Caching Proxy." [Online]. Available: http://www.squid-cache.org/

[37] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/merge: System support for elastic execution in virtual middleboxes," in *Proc. of USENIX NSDI*, Lombard, IL, USA, 2013.

[38] Y. Minlan, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Mar. 2008.

[39] Y. Zhu. and M. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," in *Proc. of IEEE INFOCOM*, Barcelona, Spain, April 2006.

[40] 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.

[41] "Ryu SDN Framework." [Online]. Available: http://osrg.github.io/ryu/

[42] 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.

[43] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *Proc. of IEEE INFOCOM*, San Francisco, California, USA, 2003.

Roberto Riggio Dr. Roberto Riggio is currently Senior Researcher staff member in the Future Networks team at CREATE-NET, Italy, and Guest Lecturer at the University of Trento, Italy. He received a PhD in Communications Engineering from University of Trento, Italy, in 2008. His research interests include performance isolation in multi-tenant mobile networks; software defined mobile networking and deep programmable mobile networks. He was, or is, involved in several European Projects including three 5G-PPP projects and has directly generated more than 1 M€ in competitive funding. Dr. Riggio has published more than 60 papers in internationally refereed journals and conferences. He serves in the TPC of leading conferences in the networking field and is associate editor for the Wiley International journal of Network Management and for the Spring Wireless Networks journal.

Abbas Bradai Dr. Abbas Bradai received his M.S in Computer Science from National Institute of Computer Science (ESI ex-INI), Algiers, Algeria, and from university of Rennes 1, France in 2009, and his PhD at LaBRI/University of Bordeaux, France, in 2012. He is actually associate professor at university of Poitiers and research fellow at XLIM lab, Poitiers. His main research interests are multimedia communications over wired and wireless networks, cognitive radio, software defined network and virtualization. Abbas Bradai is/was involved in many French and European projects (FP7, H2020) such as ENVISION and VITAL.

Davit Harutyunyan Mr. Davit Harutyunyan is a Ph.D. student at the University of Trento, Italy, working under the supervision of Prof. Imrich Chlamtac and Dr. Roberto Riggio. He received the first level Degree in Telecommunication Engineering in June 2011 and the master Degree in Telecommunication Engineering in September 2015, both with honors, from the National Polytechnic University of Armenia. Since 2015, he is a member of the Future Networks team at CREATE-NET. His main research interests include cellular networks and software defined mobile networking.

Tinku Rasheed Dr. Tinku Rasheed is a Senior Research staff member, heading the Future Networks Area within CREATE-NET. Before joining CREATE-NET, Dr. Rasheed was research engineer with Orange Labs for a period of 4 years. He received his Ph.D. degree from the Computer Science Department of the University of Paris-Sud XI., in 2007, M.S. degree in 2003 from Aston University, U.K. and bachelor degree in 2002 in Telecommunications from University of Kerala, India. Dr. Rasheed has extensive industrial and academic research experience in the mobile wireless communication area, end-to-end network architectures and services. He has several granted patents and has published more than 60 articles in major journals and conferences. He is a member of the IEEE and ACM.

Toufik Ahmed Prof. Toufik Ahmed is a full professor at ENSEIRB-MATMECA school of engineers in Institut Polytechnique de Bordeaux (Bordeaux-INP) and performing research activities in CNRS-LaBRI Lab-UMR 5800 at University Bordeaux. T. Ahmeds main research activities concern end-to-end Quality of Service (QoS) management and provisioning for multimedia wired and wireless networks, media streaming, and cross-layer optimization. T. Ahmed has also worked on a number of national and international projects. He is serving as TPC member for international conferences and journals.