IFIP Networking 2019 1570512106

Composing 5G Network Slices by Co-locating VNFs in μslices

Matteo Pozza*, Akanksha Patel†, Ashwin Rao*, Hannu Flinck‡, and Sasu Tarkoma*
*University of Helsinki, †Indian Institute of Technology Bombay, ‡Nokia Bell Labs

***Abstract*—5G networks leverage network slices for serving use cases with different and potentially conflicting requirements. Current approaches for composing network slices largely overlook the presence of multiple ingress and egress nodes in each network slice. This results in inefficient resource usage even when co-locating the Virtual Network Functions (VNFs) of a slice. The network is thus quickly saturated, and no more use cases can be served. We address this issue by introducing μslices, where each μslice is tied to a specific pair of ingress and egress nodes of a slice. Then, we optimize the bandwidth consumed by each μslice by co-locating its VNF instances that communicate the most. We observed that enabling μslicing was able to save about two times more control and data plane traffic and it also resulted in a more even distribution of computational and link load.**

I. Introduction

5G networks are expected to serve a wide range of use cases from verticals such as automotive and e-health. To satisfy the requirements from these use cases, mobile operators and equipment manufacturers envision using Virtual Network Functions (VNFs) and network slicing [1]. VNFs are software implementations of network functions that can be migrated and instantiated on demand. Network slicing prescribes multiplexing several virtual networks that are isolated from each other on a single physical network.

Composing network slices, *i.e.* specifying how to create network slices by taking as input the physical substrate and the VNFs, is non-trivial. As discussed in §II, this problem falls in the realm of VNF Forwarding Graph Embedding (VNF-FGE). Nevertheless, 5G networks distribute traffic on different paths of User Plane Functions (UPFs) to achieve scalability [2], which results in network slices having their traffic flowing through multiple pairs of ingress and egress nodes. Current solutions for composing network slices largely overlook the existence of multiple ingress and egress nodes within a network slice, thus leading to inefficient resource usage, *e.g.* see §III. As a result, the network resources are quickly saturated, and subsequent requests for new slices by other use cases must be rejected.

To address this problem, we propose to compose 5G network slices by splitting them into μslices. Each μslice consists of a group of VNF instances tied to a pair of ingress and egress nodes. In particular, each μslice serves flows traversing a specific ingress node. For each network slice, multiple instances of its VNFs are created such that each instance of a VNF is assigned to a specific μslice. This ensures that VNF instances serving different μslices are isolated from each other. Then, we optimize the bandwidth consumed by each μslice by co-locating its VNF instances that communicate the most. Note that the term μslice has been used previously by Leconte *et al.* [3] to describe a subset of all the traffic flows that are tied to a single pair of ingress and egress nodes. This description of a μslice is clearly different from ours.

Our key contributions are as follows.

- We present a novel approach for composing 5G network slices by co-locating VNFs in μslices. The key differences of our approach from existing approaches are that a) it builds on the ability to decompose a VNF into VNF instances tied to a pair of ingress and egress nodes, and b) it optimizes the bandwidth consumed by co-locating VNF instances serving a μslice.
- We present an integer linear programming model to compose network slices that takes as inputs the properties of the physical substrate and the requirements of use cases. We implement our model in an off-the-shelf solver, and our code is publicly available.[1]
- We highlight the benefits of μslicing by comparing the solutions obtained with μslicing and those obtained without μslicing across different configurations. We created these configurations by integrating data from publicly available datasets with measurements conducted in our test network. We observed that across all configurations μslicing was able to save about two times more control plane and data plane traffic. Furthermore, we observed that μslicing allowed us to find a feasible solution for problem instances declared otherwise infeasible by the solver. μslicing also resulted in a more even distribution of the computational and link load.

In the following, we discuss the related works (§II) and we define the concept of μslicing (§III). We then present our model (§IV) and some refinements to reduce complexity (§V). Finally, we discuss our evaluation (§VI), and we analyze limitations and follow-up work of our study (§VII).

II. Related Works

Composing networks by mapping virtual entities, such as VNFs and virtual links, onto a physical substrate falls into the realm of Virtual Network Embedding (VNE) and VNF Forwarding Graph Embedding (VNF-FGE). Solutions for VNE mainly focus on mapping virtual links onto the physical links. In contrast, solutions for VNF-FGE focus on

ISBN 978-3-903176-16-4

[1] https://version.helsinki.fi/matteo.pozza/avatarifip19

TABLE I
COMPARISON WITH RELATED WORKS

Work	VNF Group	VNF Co-location	Key Difference(s)
Basta *et al.* [4]	*Ingress*	Implicit	Ingress node not shareable among VNF groups
Baumgartner *et al.* [5], [6]	*Ingress*	Implicit	Ingress node not shareable among VNF groups
Bhamare *et al.* [7]	FC	Implicit	Scheduler for Function Chain requests
Carpio *et al.* [8]	FC	ND	OBJ: load balancing
Eramo *et al.* [9]	ND	*Explicit*	OBJ: minimize amount of traffic rejected
Kuo *et al.* [10]	ND	Implicit	OBJ: maximize number of accepted Function Chain requests
Kuo *et al.* [11]	FC	ND	OBJ: maximize number of flows embedded
Leconte *et al.* [3]	*Ingress*	ND	Single pair of ingress and egress nodes
Liu *et al.* [12]	ND	ND	OBJ: maximize profit
Martini *et al.* [13]	FC	Implicit	OBJ: minimize communication latency
Mehraghdam *et al.* [14]	ND	*Explicit*	No VNF grouping; quadratic model
Meng *et al.* [15]	ND	*Explicit*	No VNF grouping; quadratic model
Palkar *et al.* [16]	ND	*Explicit*	Rough specification of demands
Patel *et al.* [17]	*Ingress*	Implicit	OBJ: minimize handover latency
Pham *et al.* [18]	FC	*Explicit*	OBJ: trade-off between traffic cost and load on nodes
Song *et al.* [19]	ND	Implicit	OBJ: trade-off between traffic cost and load on nodes
Taleb *et al.* [20]	ND	ND	OBJ: minimize (UE, P-GW) path and minimize S-GW relocations
Wen *et al.* [21]	ND	*Explicit*	OBJ: minimize number of VNFs deployed
Ye *et al.* [22]	FC	*Explicit*	VNFs not shared between Function Chains of same use case
Our approach	*Ingress*	*Explicit*	OBJ: maximize traffic between co-located VNFs

composing networks where VNFs are organized into Function Chains. Note that *each Function Chain defines the order in which traffic traverses through its VNFs*. In 5G networks, the VNFs are traversed in a specific order during procedures such as registration and deregistration. For instance, the order of the VNFs traversed during a registration procedure is different from the order followed during deregistration [23]. Composing 5G slices therefore falls into the VNF-FGE category.

In Table I, we summarize the key related works on VNF-FGE; other works on VNE and VNF-FGE are summarized in the surveys by Fischer *et al.* [24] and Herrera *et al.* [25]. We compare the related works using two parameters: approach for VNF grouping, and support for VNF co-location.

The approach for VNF grouping determines the granularity with which a solution can create groups of VNFs such that VNFs of one group are isolated from those in other groups. The ability to group VNFs is a key property for network slicing because network slices are isolated from each other. The solution may either a) not discuss the grouping approach (ND), b) group VNFs according to the Function Chain they serve (FC), or c) group VNFs serving flows traversing a specific ingress node of the network (Ingress). Grouping by Function Chain is not suitable for network slicing because the VNF instances serving a network slice might be shared by several procedures. For example, an instance of the Access and Mobility Function (AMF) is expected to be used by both registration and deregistration procedures. Instead, grouping by ingress node is desirable because it allows sharing of VNFs between the different procedures originating from the same ingress node. Our analysis shows that VNF grouping on a per-ingress-node basis has been explored only by few works.

VNF co-location optimizes link utilization by placing VNFs that communicate the most on the same physical node. We observe that VNF co-location is either a) not discussed (ND), b) implicitly achieved as a side effect (Implicit), or c) explicitly specified as objective (Explicit). To achieve a more efficient resource usage, it is desirable to explicitly specify VNF co-location as objective.

Among the works that group VNFs according to ingress nodes, the works of Baumgartner *et al.* [5], [6] and Basta *et al.* [4] do not allow sharing the same ingress node in the physical substrate between two independent groups of VNFs. Sharing an ingress node is a key property for network slicing because traffic from different use cases might be originated at the same ingress node. Moreover such works do not deal with mobility requirements of the users of each use case. In contrast, Patel *et al.* [17] deal with mobility requirements, however the objective of their work is to minimize the handover latency. The recent work of Leconte *et al.* [3] introduces the concept of μslices, which the authors leverage to split the traffic between a pair of ingress and egress nodes in sub-flows. However, their work considers only a single pair of ingress and egress nodes.

III. OVERVIEW OF μSLICING

A network slice has a set of VNFs and a corresponding set of procedures describing how the VNFs communicate with each other. Furthermore, VNFs of one slice do not communicate with VNFs of another slice because network slices are expected to be isolated from each other.

In our model, each VNF of a network slice is labeled as either a) *ingress*, b) *egress*, or c) *intermediate*. An *ingress* VNF is the first VNF traversed by traffic entering the network, while an *egress* VNF is the last VNF traversed before exiting the network. All other VNFs are labeled as *intermediate* VNFs. Each network slice has exactly one ingress VNF and exactly one egress VNF. Note that there can be multiple instances of the same VNF simultaneously running in the network. A VNF can also have an additional label of being a *mobility anchor*, *i.e.* an *ingress*, *egress*, or *intermediate* VNF can also be a *mobility anchor*. A VNF is a *mobility anchor* if two instances of this VNF exchange user traffic when a user relocates to another ingress node.

We assume that a 5G network assigns one network slice for each use case; this mapping of use cases to slices is in line with the various ways in which network slices can be created [26]. Each use case specifies a set of node pairs, where each pair consists of an ingress node and the corresponding egress node. Pairs of the same use case have different ingress nodes, while an egress node may be shared by multiple pairs.

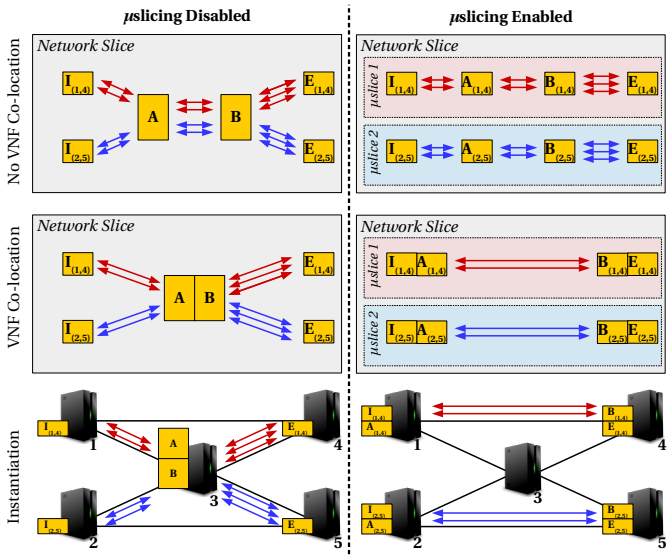

Fig. 1. **Composing a network slice by co-locating VNFs when μslicing is disabled, and when μslicing is enabled.** We assume a network slice with i) four VNFs: I, A, B, and E, and ii) fives nodes: 1, 2, 3, 4, and 5. Nodes 1 and 2 are ingress nodes, while nodes 4 and 5 are the corresponding egress nodes. The number of arrows indicate the amount of communication between a pair of VNFs. A μslice is labeled by the ingress-egress pair, *i.e.* (1,4) and (2,5). VNF I is an ingress VNF, *i.e.* a VNF which must be instantiated on an ingress node. Similarly, VNF E is an egress VNF. VNFs A and B are intermediate VNFs which can be instantiated anywhere in the network. In this example, μslicing can provide traffic savings and distribute the load more evenly.

Consequently, each pair is identified within a use case by its ingress node. Note that different use cases may use the same ingress and egress nodes. For each use case, its ingress and egress VNFs must be instantiated on an ingress and egress nodes respectively.

The idea behind μslicing is to decompose intermediate VNFs into instances that are tied to a pair of ingress and egress nodes. The μslices are isolated from each other, except for the communication between mobility anchors. In Figure 1, we present an example network slice with one ingress VNF, I, two intermediate VNFs, A and B, and one egress VNF, E. This slice has two pairs of ingress and egress nodes: (1,4) and (2,5). When μslicing is disabled and VNFs that communicate the most are co-located, we observe that the intermediate VNFs can be co-located and instantiated on node 3. Instead, when μslicing is enabled, the network slice is split into two μslices. Each μslice has its own instances of intermediate VNFs that are tied to the corresponding pair of ingress and egress nodes. As shown in Figure 1, co-locating VNF instances in μslices is more effective in optimizing resources of the physical substrate.

IV. System Model

We now present an integer programming model whose objective function maximizes the amount of traffic savings by co-locating VNFs in μslices. Table II provides a short description of the model input.

TABLE II
Model Input Description

Physical Substrate	
N	set of nodes
L	set of links
$capN_n$	processing capacity of node n (ops/s)
$capL_l$	capacity of link l (Bps)
$latL_l$	latency of link l (s)
$edge_{n_1,n_2,l}$	1, if link l connects node n_1 to n_2; 0, otherwise
$path_{n_1,n_2,l}$	1, if path from node n_1 to node n_2 includes link l; 0, otherwise
Requirements of Use Cases	
U	set of use cases
V	set of VNFs
P	set of procedures
$mapU_{u,v}$	1, if VNF v serves use case u; 0, otherwise
$ingV_v$	1, if VNF v is an ingress VNF; 0, otherwise
$egrV_v$	1, if VNF v is an egress VNF; 0, otherwise
$flocV_{n,s,v}$	1, if VNF v serving μslice s is instantiated at node n; 0, otherwise
$vopsP_{p,v}$	operations required by an instance of VNF v to serve a procedure p (ops)
$vcomP_{p,v_1,v_2}$	bytes exchanged between instances of VNFs v_1 and v_2 to serve a procedure p (B)
$reqP_{p,s}$	rate of requests of procedure p for μslice s (Hz)
$mdelP_p$	max acceptable delay for completing procedure p (s)
Intermediate Calculations	
$IvcomP_{p,v_1,v_2}$	1, if $vcomP_{p,v_1,v_2} > 0$; 0, otherwise
$isS_{s,u}$	1, if s identifies a μslice for use case u; 0, otherwise
$procS_{s,v}$	total processing capacity required by an instance of VNF v when serving μslice s (ops/s)
$comS_{s,v_1,v_2}$	total bandwidth required between instances of VNFs v_1 and v_2 when serving μslice s (Bps)
$pathDel_{n_1,n_2}$	total delay between nodes n_1 and n_2 (s)

A. Physical Substrate

Our physical substrate corresponds to a connected graph having a set N of nodes and a set L of links. Each node n has a processing capacity $capN_n$, while each link l has a capacity $capL_l$ and a latency $latL_l$. For each link l and each pair of nodes n_1 and n_2, $edge_{n_1,n_2,l}$ indicates if link l connects node n_1 to node n_2, and $path_{n_1,n_2,l}$ indicates if the communication path from node n_1 to node n_2 traverses link l.

B. Requirements of Use Cases

The set U contains all the use cases served by the network. Each use case is assigned one network slice, so there are $|U|$ network slices to compose in the network. The sets V and P respectively contain all VNFs and all procedures required to serve the use cases in U. The mapping between use cases and VNFs is given by $mapU_{u,v}$, where $mapU_{u,v} = 1$ if VNF v serves use case u. Given a VNF v, $ingV_v$ and $egrV_v$ indicate respectively if v is an ingress VNF and if v is an egress VNFs.

As described in §III, we decompose each network slice into μslices such that each μslice consists of a group of VNF instances tied to a pair of ingress and egress nodes. Since each pair is identified within a network slice by its ingress node, we use the ingress node to identify the corresponding μslice. Given a node n, a μslice s, and a VNF v, $flocV_{n,s,v}$ indicates if VNF v serving μslice s must be instantiated at node n. $flocV$ is thus used to indicate requirements on the location of the VNF instances, *e.g.* ingress and egress VNFs placed respectively at the ingress and egress nodes.

Each procedure requires a subset of VNFs of a slice to communicate with each other, and it also incurs a computational load on those VNFs. We use $vopsP_{p,v}$ to specify the number of operations required by an instance of VNF v to serve a procedure p, and $vcomP_{p,v_1,v_2}$ to specify the amount of bytes exchanged between instances of VNFs v_1 and v_2 to serve a procedure p. Note that there is no communication between two instances of the same VNF ($vcomP_{p,v,v}$ = 0), except when considering mobility procedures, *e.g.* handover, because data of the moving user is forwarded from the mobility anchor VNF of the source μslice to the corresponding one in the target μslice. In addition, we use $reqP_{p,s}$ to specify the rate of requests of procedure p for μslice s. Note that the above variables are set to 0 when VNFs, procedures, and μslices are not related with the same use case. For example, if procedure p is tied to a network slice that has no μslice s, then $reqP_{p,s}$ is set to 0. Finally, we use $mdelP_p$ to indicate the maximum acceptable delay for completing procedure p.

C. Intermediate Calculations

To succinctly represent some key aspects of the model input, we introduce the following parameters. These parameters are computed by pre-processing the model input, thus they do not affect the complexity of the model.

Given a procedure p and two VNFs v_1 and v_2, the binary flag $IvcomP_{p,v_1,v_2}$ indicates if the VNFs communicate with each other during the procedure. We define it as follows:

$$IvcomP_{p,v_1,v_2} = \begin{cases} 1 & \text{if } vcomP_{p,v_1,v_2} > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Given a use case u and a node s, we use $isS_{s,u}$ to indicate if use case u declares s as an ingress node, *i.e.* if s identifies a μslice of use case u. We define it as follows:

$$isS_{s,u} = \sum_{v \in V} mapU_{u,v} \cdot ingV_v \cdot flocV_{s,s,v}, \quad \forall s \in N, u \in U. \quad (2)$$

To exemplify the use of isS, given a use case u and a node s, the node s is an ingress node of use case u ($isS_{s,u}$ = 1) if there exists a VNF v such that a) v serves use case u ($mapU_{u,v} = 1$), b) v is an ingress VNF ($ingV_v = 1$), and c) v is instantiated in the node s to serve μslice s ($flocV_{s,s,v} = 1$).

The total processing capacity required by an instance of VNF v when serving μslice s is given by $procS_{s,v}$, where

$$procS_{s,v} = \sum_{p \in P} vopsP_{p,v} \cdot reqP_{p,s}, \forall s \in N, v \in V. \quad (3)$$

Similarly, the total bandwidth required between instances of VNFs v_1 and v_2 when serving μslice s is given by $comS_{s,v_1,v_2}$, where

$$comS_{s,v_1,v_2} = \sum_{p \in P} vcomP_{p,v_1,v_2} \cdot reqP_{p,s}, \quad \forall s \in N, v_1, v_2 \in V. \quad (4)$$

Note that the two previous calculations result in 0 when a VNF is not used by a use case. For example, if a use case u declares s as ingress node ($ingN_{s,u} = 1$), and consequently there is μslice s in the corresponding network slice, but VNF v does not serve use case u ($mapU_{u,v} = 0$), then $procS_{s,v}$ is 0.

Finally, the total delay between a pair of nodes is given by $pathDel_{n_1,n_2}$, where

$$pathDel_{n_1,n_2} = \sum_{l \in L} path_{n_1,n_2,l} \cdot latL_l, \forall n_1, n_2 \in N. \quad (5)$$

D. Integer Programming Model

The objective of our model is to optimize the amount of bandwidth used in the links of the physical substrate by co-locating VNF instances in μslices. We introduce the following binary decision variables:

$$locV_{n,s,v} = \begin{cases} 1 & \text{if VNF } v \text{ serving } \mu\text{slice } s \\ & \quad \text{is instantiated at node } n \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Specifically, our objective is to maximize the traffic between VNF instances that are co-located in the same node of the physical substrate because such traffic does not leave the co-location node. As described in §III, instances of VNFs serving different μslices do not communicate because μslices are isolated from each other. The only exception to this rule is in case of instances of a mobility anchor VNF serving different μslices because mobility procedures require them to exchange traffic related with moving users. As a result, the model maximizes a) the traffic between co-located instances of different VNFs serving the same μslice, and b) the traffic between co-located instances of the same mobility anchor VNF serving different μslices. This objective is expressed using the following objective function.

$$maximize: \sum_{\substack{s \in N \\ v_1, v_2 \in V \\ v_1 \neq v_2}} comS_{s,v_1,v_2} \cdot \left(\sum_{n \in N} locV_{n,s,v_1} \cdot locV_{n,s,v_2} \right) + \sum_{\substack{s_1 \in N \\ v \in V}} comS_{s_1,v,v} \cdot \left(\sum_{\substack{n, s_2 \in N \\ s_1 \neq s_2}} locV_{n,s_1,v} \cdot locV_{n,s_2,v} \right) \quad (7)$$

The constraints of our model are as follows.

1) Instantiation constraints force to instantiate a VNF serving a μslice in at most one node. They are expressed as

$$\sum_{n \in N} locV_{n,s,v} \leq N \cdot (1 - mapU_{u,v}) + mapU_{u,v} \cdot isS_{s,u}$$
$$\sum_{n \in N} locV_{n,s,v} \geq mapU_{u,v} \cdot isS_{s,u} \quad (8)$$
$$\forall s \in N,\ v \in V,\ u \in U.$$

2) Fixed VNF location constraints ensure that a VNF whose location is specified through $flocV$ to a certain node is not instantiated in another node. They are expressed as

$$flocV_{n,s,v} \leq locV_{n,s,v}, \quad \forall n, s \in N,\ v \in V. \quad (9)$$

3) Processing capacity constraints ensure that the processing capacity of nodes is not exceeded. They are expressed as

$$\sum_{s \in N,\ v \in V} procS_{s,v} \cdot locV_{n,s,v} \leq capN_n, \quad \forall n \in N. \tag{10}$$

4) Link capacity constraints ensure that the capacity of the links is not exceeded. They are expressed as

$$\begin{aligned} &\sum_{\substack{s,n_1,n_2 \in N \\ v_1,v_2 \in V}} comS_{s,v_1,v_2} \cdot path_{n_1,n_2,l} \cdot locV_{n_1,s,v_1} \cdot locV_{n_2,s,v_2} \\ &+ \sum_{\substack{s_1,s_2,n_1,n_2 \in N \\ v \in V}} comS_{s_1,v,v} \cdot path_{n_1,n_2,l} \cdot locV_{n_1,s_1,v} \cdot locV_{n_2,s_2,v} \\ &\leq capL_l, \quad \forall l \in L. \end{aligned} \tag{11}$$

5) Procedure delay constraints ensure that the time required to complete the procedures does not exceed their maximum acceptable delay. They are expressed as

$$\begin{aligned} &\sum_{\substack{n_1,n_2 \in N \\ v_1,v_2 \in V}} pathDel_{n_1,n_2} \cdot IvcomP_{p,v_1,v_2} \cdot locV_{n_1,s,v_1} \cdot locV_{n_2,s,v_2} \\ &+ \sum_{\substack{s_2,n_1,n_2 \in N \\ v \in V}} pathDel_{n_1,n_2} \cdot IvcomP_{p,v,v} \cdot locV_{n_1,s,v} \cdot locV_{n_2,s_2,v} \\ &\leq mdelP_p, \quad \forall s \in N,\ p \in P. \end{aligned} \tag{12}$$

Note that link capacity and procedure delay constraints have the same structure as the objective function. Indeed, the first row deals with instances of different VNFs serving the same μslice, and the second row deals with instances of the same VNF serving different μslices.

V. Complexity Optimizations

We now describe the optimizations we have performed to reduce the complexity of our model.

A. Achieving Linearity

The proposed model has both a quadratic objective function and quadratic constraints, which make it difficult to solve by off-the-shelf solvers. To linearize the model, we introduce decision variables $locVPair$ defined as follows:

$$locVPair^{n_1,s_1,v_1}_{n_2,s_2,v_2} = \begin{cases} 1 & \text{if } locV_{n_1,s_1,v_1} \wedge locV_{n_2,s_2,v_2} = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

We add the following constraints to model each $locVPair$ variable as a binary conjunction of two $locV$ variables.

$$\begin{aligned} locVPair^{n_1,s_1,v_1}_{n_2,s_2,v_2} &\leq locV_{n_1,s_1,v_1} \\ locVPair^{n_1,s_1,v_1}_{n_2,s_2,v_2} &\leq locV_{n_2,s_2,v_2} \\ locV_{n_1,s_1,v_1} + locV_{n_2,s_2,v_2} - locVPair^{n_1,s_1,v_1}_{n_2,s_2,v_2} &\leq 1 \\ \forall s_1,n_1,s_2,n_2 \in N,\ v_1,v_2 \in V & \end{aligned} \tag{14}$$

We use $locVPair$ variables when we have a multiplication between $locV$ variables in our model. The objective function is now expressed as follows.

$$\begin{aligned} &maximize: \\ &\sum_{\substack{s \in N \\ v_1,v_2 \in V \\ v_1 \neq v_2}} comS_{s,v_1,v_2} \cdot \left(\sum_{n \in N} locVPair^{n,s,v_1}_{n,s,v_2} \right) + \\ &\sum_{\substack{s_1 \in N \\ v \in V}} comS_{s_1,v,v} \cdot \left(\sum_{\substack{n,s_2 \in N \\ s_1 \neq s_2}} locVPair^{n,s_1,v}_{n,s_2,v} \right) \end{aligned} \tag{15}$$

The link capacity constraints are now expressed as

$$\begin{aligned} &\sum_{\substack{s,n_1,n_2 \in N \\ v_1,v_2 \in V}} comS_{s,v_1,v_2} \cdot path_{n_1,n_2,l} \cdot locVPair^{n_1,s,v_1}_{n_2,s,v_2} \\ &+ \sum_{\substack{s_1,s_2,n_1,n_2 \in N \\ v \in V}} comS_{s_1,v,v} \cdot path_{n_1,n_2,l} \cdot locVPair^{n_1,s_1,v}_{n_2,s_2,v} \\ &\leq capL_l, \quad \forall l \in L. \end{aligned} \tag{16}$$

The procedure delay constraints are now expressed as

$$\begin{aligned} &\sum_{\substack{n_1,n_2 \in N \\ v_1,v_2 \in V}} pathDel_{n_1,n_2} \cdot IvcomP_{p,v_1,v_2} \cdot locVPair^{n_1,s,v_1}_{n_2,s,v_2} \\ &+ \sum_{\substack{s_2,n_1,n_2 \in N \\ v \in V}} pathDel_{n_1,n_2} \cdot IvcomP_{p,v,v} \cdot locVPair^{n_1,s,v}_{n_2,s_2,v} \\ &\leq mdelP_p, \quad \forall s \in N,\ p \in P. \end{aligned} \tag{17}$$

B. Reducing the Number of Decision Variables

A drawback of introducing $locVPair$ variables is that it brings $(|N| \cdot |N| \cdot |V|)^2$ more variables to the solver, and this becomes intractable for large values of $|N|$ or $|V|$.

One can note that the instantiation constraints (8) force $locV_{n,s,v} = 0, \forall n \in N$ if v serves use case u but u does not declare s as an ingress node. As a consequence, $locV$ variables satisfying such condition do not influence the value of the objective function nor affect the satisfiability of the other constraints, and therefore they can be removed from the problem definition. We thus define a parameter def as follows:

$$def_{n,s,v} = \begin{cases} 1 & \text{if } \exists\, u \in U \,|\, mapU_{u,v} \wedge isS_{s,u} = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

When implementing the problem in the solver, we define $locV_{n,s,v}$ if $def_{n,s,v} = 1$. Consequently, we define $locVPair^{n_1,s_1,v_1}_{n_2,s_2,v_2}$ if $def_{n_1,s_1,v_1} = 1$ and $def_{n_2,s_2,v_2} = 1$, thus allowing to reduce significantly the number of decision variables defined in the solver.

In summary, we show that it is indeed possible to linearize the model and reduce its number of decision variables when implementing it in the solver.

VI. Numerical Results

We now detail the goal of our evaluation (§VI-A), the scenarios considered (§VI-B), and the results obtained (§VI-C).

A. Goal

Our aim is to assess the benefits brought by μslices, *i.e.* how μslices make more efficient the co-location of VNFs when composing network slices. We therefore compare the results obtained by enabling μslicing with those obtained by disabling μslicing considering the same experimental setup. We leave the comparison of our approach with other network slices composition approaches as future work.

We disable μslicing by adding the following constraints,

$$\begin{aligned}(1 - ingV_v) \cdot (1 - egrV_v) \cdot mapU_{u,v} \cdot isS_{s_1,u} \\ \cdot isS_{s_2,u} \cdot (locV_{n,s_1,v} - locV_{n,s_2,v}) = 0, \\ \forall n, s_1, s_2 \in N,\ v \in V,\ u \in U.\end{aligned} \tag{19}$$

The above constraints ensure that the instances of a VNF v mapped to a use case u are all instantiated in the same node of the physical substrate; the only exceptions are VNFs which must be instantiated on ingress or egress nodes. Note that a model which uses the above constraints along with the constraints and objective function defined in §IV is still aimed at finding solutions where VNFs that communicate the most are co-located, albeit with μslicing disabled. For example, it searches for solutions such as the one presented in Figure 1 where VNFs A and B are co-located.

B. Scenarios

We now describe the information we gathered from technical documents and measurements to build realistic scenarios.

1) Physical Substrate: We used three real network instances for our evaluation: New York, from SNDlib [27], and Beijing and Tokyo, from the Internet Topology Zoo [28]. We pruned the topology graphs by removing disconnected nodes and duplicate links. Consequently, the New York topology graph had 16 nodes and 49 edges, while Beijing and Tokyo topology graphs had 9 nodes and 10 edges, and 12 nodes and 13 edges respectively. We assume each node in the network corresponds to a cluster equipped with 100 servers, and the links between these nodes to be point-to-point fiber links with 10 Gbps of bandwidth and 1 ms of latency [29]. The processing capacity of each server was set using the highest value in the results of the Standard Performance Evaluation Corporation (SPEC) benchmark suite [30]. We assume that each pair of non-adjacent nodes communicate using the path having the smallest number of hops. To account for hardware improvements in the years to come, we also consider scenarios in which the number of servers at each node and the bandwidth of each link is increased by a multiplying factor. Specifically, we consider multiplying factors (x1, x2.5, x5, x7.5, x10) for both number of servers (p_1, p_2, p_3, p_4, p_5) and link bandwidth (c_1, c_2, c_3, c_4, c_5) and we consider all 25 combinations of such factors. As an example, the combination (p_3, c_4) means that each node has $5 * 100 = 500$ servers and each link has $7.5 * 10 = 75$ Gbps of bandwidth.

TABLE III
Size of signals during an Active-to-Idle (AtI) transition.

Name	Direction	Size (B)
UE Context Release Request	eNB → MME	90
Release Access Bearers Request	MME → S-GW	54
Release Access Bearers Response	S-GW → MME	60
UE Context Release Command	MME → eNB	102
UE Context Release Complete	eNB → MME	98

2) Requirements of Use Cases: We used the VNFs and procedures of LTE because of the limited information on how use cases will use the VNF and procedures of 5G. Specifically, we consider the following four control plane procedures: Initial Attach (IA), Active-to-Idle (AtI) transition, Idle-to-Active (ItA) transition, and X2 handover. We abstract data downloads and uploads as data plane procedures. A download procedure consists of two types of signals, one from the Packet Data Network Gateway (P-GW) to the Serving Gateway (S-GW) followed by another from the S-GW to the evolved Node B (eNB); uploads use the same signals in the reverse direction.

In our evaluation, the number of operations required by a VNF to complete a procedure corresponds to the number of signals received by the VNF during that procedure. We obtain these numbers by analyzing the LTE specifications [31] and the traffic between VNFs when running an instance of an LTE core in our test network. For instance, Table III summarizes the signals exchanged between VNFs during an Active-to-Idle (AtI) transition. We create similar tables for the IA, ItA transition, and X2 handover procedures, however we do not present them here due to space constraints. To estimate the data plane traffic going from a Source eNB (SeNB) to a Target eNB (TeNB) during an X2 handover, we measured the number of bytes traversing the X2 link. We downloaded a 1.5 GB file on a mobile phone connected to a SeNB and then moved in the direction of a TeNB, and we repeated the process 10 times. We obtained an average of approximately 120 KB of data packets transferred from SeNB to TeNB.

We defined two use cases, UC1 and UC2, by leveraging the concept of quality class in LTE which is identified by a Quality Class Identifier (QCI). For UC1 we select live streaming of conversational video (QCI 2), and for UC2 we select buffered streaming of video (QCI 8). We use these quality classes to represent use cases having different requirements from the infrastructure. For our evaluation, the underlying infrastructure hosts two network slices, one for UC1 and the other for UC2.

We consider five pairs of ingress and egress nodes for each use case. For choosing these nodes, we begin by randomly selecting one ingress node in the network. We then select the other ingress nodes randomly from its adjacent nodes. If there are not enough adjacent nodes we select nodes adjacent to the ones already selected till we get five ingress nodes. This selection process creates a group of adjacent ingress nodes so that X2 handovers are allowed between each pair of nodes in the group. Finally, for each ingress node, we choose its egress node by selecting a random node in the network.

For each use case, we assume 1000 users at each ingress

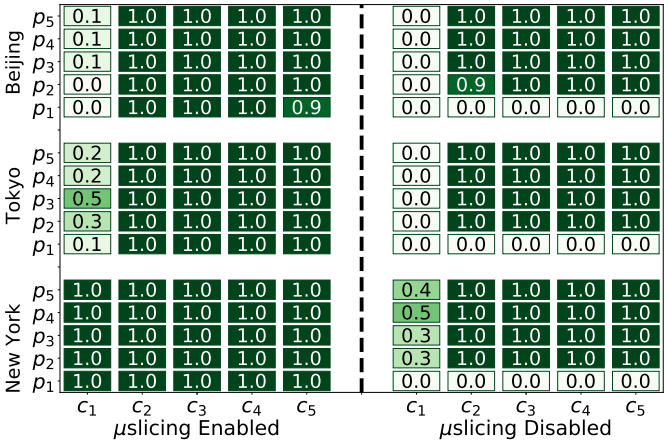

Fig. 2. **Feasibility ratio for each configuration.**

TABLE IV
RATIO OF AVERAGE TRAFFIC SAVINGS $\frac{\mu\text{SLICING ENABLED}}{\mu\text{SLICING DISABLED}}$
(CONTROL PLANE, DATA PLANE)

Topology	Num Servers	Link Bandwidth				
		c_1	c_2	c_3	c_4	c_5
Beijing	p_5	↑ , ↑	2.68, 2.99	2.36, 1.98	2.21, 2.12	2.43, 2.35
	p_4	↑ , ↑	2.34, 2.57	2.28, 2.32	2.28, 2.23	2.27, 2.25
	p_3	↑ , ↑	2.41, 2.50	2.26, 2.02	2.27, 2.16	2.23, 1.98
	p_2	— , —	2.36, 2.71	2.29, 2.24	2.37, 2.24	2.34, 2.24
	p_1	— , —	↑ , ↑	↑ , ↑	↑ , ↑	↑ , ↑
Tokyo	p_5	↑ , ↑	2.63, 2.70	2.34, 2.36	2.22, 2.23	2.30, 2.33
	p_4	↑ , ↑	2.46, 2.99	2.29, 2.44	2.24, 2.35	2.26, 2.37
	p_3	↑ , ↑	2.49, 2.58	2.26, 2.30	2.34, 2.29	2.23, 2.25
	p_2	↑ , ↑	2.40, 2.88	2.33, 2.24	2.20, 2.36	2.30, 2.40
	p_1	↑ , ↑	↑ , ↑	↑ , ↑	↑ , ↑	↑ , ↑
New York	p_5	2.40, 4.11	2.39, 2.57	2.34, 2.53	2.45, 2.52	2.26, 2.51
	p_4	2.22, 2.80	2.30, 2.46	2.42, 2.26	2.39, 2.50	2.49, 2.59
	p_3	3.39, 3.10	2.33, 2.56	2.52, 2.64	2.31, 2.51	2.25, 2.45
	p_2	2.19, 2.75	2.37, 2.57	2.37, 2.33	2.38, 2.67	2.46, 2.43
	p_1	↑ , ↑	↑ , ↑	↑ , ↑	↑ , ↑	↑ , ↑

node, resulting in 5000 users. The number is in line with the number of users in recent 5G trials [32], although we acknowledge that in 5G networks the number of users might vary across use cases. We use the frequency of control plane procedures reported by Metsälä *et al.* [33] and Tabbane *et al.* [34] for the UC1 and UC2 respectively; we use the frequency of data plane procedures for UC1 and UC2 from a white paper from Huawei [35]. We use the maximum packet delay values provided by the 3GPP community [36] to define the delay limits on the download and upload procedures of the two use cases. We define the delay limits on control plane procedures using the delay budgets reported by Savic *et al.* [37].

C. Results

For each network instance, we consider all combinations of the multiplying factors for number of servers at each node and link bandwidth. Such triples (network instance, servers factor, bandwidth factor) are henceforth referred to as configurations. As an example, configuration (*Beijing*, p_3, c_4) refers to the Beijing physical substrate in which the number of servers for each node is 500 and the bandwidth of each link is 75 Gbps. For each configuration, we generated 10 problem instances. We implemented our model using IBM ILOG Cplex Optimization Studio 12.7.1, and for each problem instance we run the solver on a server with 32 CPU cores and 256 GB of memory. We set the solver to a maximum of 10 minutes to solve each problem instance, after which it returns the best solution found, if any. For each problem instance, we run the solver two times, *i.e.* with μslicing enabled and with μslicing disabled.

We compare the results obtained by enabling μslicing with those obtained by disabling it using the following metrics.

a) Feasibility ratio: the fraction of problem instances for which a feasible solution was found within 10 minutes.

b) Traffic savings: the amount of control plane and data plane traffic which was saved by co-locating the VNFs.

c) Resources usage: the amount of computational and bandwidth resources consumed by the VNFs.

These metrics help us quantify the bandwidth savings by μslicing and VNF co-location. Note that the delay incurred by procedures was governed by the delay constraints (12).

1) Feasibility Ratio: In Figure 2, we present the observed feasibility ratio across all configuration when a) μslicing is enabled and b) μslicing is disabled. We observe that the solver is able to find solutions when μslicing is enabled even in configurations where no solutions were found when μslicing was disabled. This is evident in constrained configurations, *i.e.* configurations with either p_1 or c_1, and the benefits of μslicing are clearly evident in configurations with p_1. This is in line with our expectations because μslicing allows us to split the load of a single VNF in smaller VNF instances that can be distributed across several nodes.

2) Traffic Savings: For each problem instance with a feasible solution, we computed the amount of control plane and data plane traffic saved by VNF co-location. This savings is measured as the amount of control plane and data plane traffic exchanged by VNFs instantiated on the same physical node. We then compute the average savings across the solved instances with the same configuration.

In Table IV, we present the ratios $i/j, x/y$ for each configuration, where i and x are respectively the average control plane and data plane traffic savings with μslicing enabled, and j and y are the corresponding values with μslicing disabled. As an example, for the configuration ($Beijing$, p_5, c_2), we observe that the average control plane traffic saved with μslicing enabled is 2.68 times more than the corresponding savings with μslicing disabled. Similarly, the average data plane traffic saved with μslicing enabled is 2.99 times more than the corresponding savings with μslicing disabled. A '—, —' for a configuration implies that the solver was not able to find solutions with μslicing enabled nor with μslicing disabled. Instead a '↑, ↑' implies that feasible solutions were found only when μslicing is enabled.

We observe that co-locating VNFs in μslices is effective in saving control plane and data plane traffic. From the control plane perspective, we observe that μslicing was able to save

TABLE V
RATIO OF STANDARD DEVIATION OF RESOURCE USAGE $\frac{\mu\text{SLICING ENABLED}}{\mu\text{SLICING DISABLED}}$
(NODE UTILIZATION, LINK UTILIZATION)

Topology	Num Servers	Link Bandwidth				
		c_1	c_2	c_3	c_4	c_5
Beijing	p_5	↑ , ↑	0.75, 0.51	0.54, 0.51	0.54, 0.48	0.44, 0.50
	p_4	↑ , ↑	0.71, 0.46	0.55, 0.51	0.58, 0.52	0.53, 0.46
	p_3	↑ , ↑	0.60, 0.51	0.47, 0.46	0.46, 0.52	0.53, 0.53
	p_2	— , —	0.71, 0.54	0.51, 0.54	0.48, 0.51	0.46, 0.54
	p_1	— , —	↑ , ↑	↑ , ↑	↑ , ↑	↑ , ↑
Tokyo	p_5	↑ , ↑	0.62, 0.59	0.45, 0.60	0.46, 0.56	0.45, 0.51
	p_4	↑ , ↑	0.63, 0.58	0.42, 0.53	0.45, 0.54	0.44, 0.56
	p_3	↑ , ↑	0.66, 0.57	0.50, 0.54	0.42, 0.54	0.42, 0.50
	p_2	↑ , ↑	0.65, 0.62	0.48, 0.56	0.52, 0.57	0.50, 0.62
	p_1	↑ , ↑	↑ , ↑	↑ , ↑	↑ , ↑	↑ , ↑
New York	p_5	0.73, 0.63	0.60, 0.56	0.59, 0.50	0.53, 0.52	0.56, 0.52
	p_4	0.79, 0.57	0.64, 0.55	0.64, 0.54	0.65, 0.55	0.62, 0.51
	p_3	0.80, 0.62	0.63, 0.56	0.53, 0.59	0.59, 0.55	0.65, 0.55
	p_2	0.78, 0.52	0.70, 0.56	0.63, 0.57	0.61, 0.57	0.59, 0.58
	p_1	↑ , ↑	↑ , ↑	↑ , ↑	↑ , ↑	↑ , ↑

at least two times the traffic across all configurations with feasible solutions. From the data plane perspective, we observe that μslicing was able to save up to 4.11 times more traffic.

3) Resources Usage: To study how computational and traffic load is distributed, we compare the standard deviation of node and link utilizations across all solved instances with the same configuration when μslicing is enabled and when μslicing is disabled. For each problem instance with a feasible solution we computed the average node and link utilizations, and their respective standard deviations. We then used these values to compute the average node utilization, the average link utilization, and the standard deviations of node and link utilizations across all the solved instances with the same configuration. Because the total computational load on the physical substrate is fixed, the average node utilization across all solved instances of a given configuration is the same when μslicing is enabled or disabled. Nevertheless, μslicing affects the distribution of the computation load, and this in turn affects the distribution of the load on the links.

In Table IV, we present the ratios $i/j, x/y$ for each configuration, where i and x are respectively the standard deviation of node and link utilizations with μslicing enabled, and j and y are the corresponding values with μslicing disabled. As an example, for the configuration ($Beijing$, p_5, c_2), we observe that the standard deviation of node utilization with μslicing enabled corresponds to 0.75 of the standard deviation with μslicing disabled, thus the computational load was spread more evenly when μslicing is enabled. Across all configurations we observe ratios smaller than 1 which imply that the load is more evenly distributed when μslicing is enabled.

We observe that the ratio of the average computational load when μslicing is enabled and the corresponding value when μslicing is disabled is close to 1 across all configurations. At the same time, we observe that the ratio of the average link load when μslicing is enabled and the corresponding load when μslicing is disabled is less than 1 across all configurations. We do not present those results due to lack of space. Instead, we present the average computational and link load with μslicing enabled and disabled for four configurations in Figure 3. The error bars indicate the minimum and maximum values across all problem instances for a given configuration where a feasible solution was found. As expected, we observe that the average node utilization is not affected by μslicing. However, the error bars show that the distribution of the computational load is skewed when μslicing is disabled. We also observe that a) the average link utilization is either smaller or equal when μslicing is enabled, and b) the distribution of the link load is skewed when μslicing is disabled.

VII. DISCUSSION AND FUTURE WORKS

We propose to decompose each network slice in μslices, each of which is tied to a pair of ingress and egress nodes of the use case served by the network slice, and to co-locate the VNFs of such μslices to save the bandwidth. We provide an integer linear programming model whose objective function maximizes the amount of traffic savings by co-locating VNFs in μslices. We implemented and evaluated it against VNF co-location with μslicing disabled, and we observed that our approach a) can provide feasible solutions when no solutions are found without μslicing, b) saves around two times more control plane and data plane traffic, and c) records a smaller standard deviation of resource utilization, implying that the load is more evenly distributed. The resource efficiency brought by μslices thus allows mobile operators to accept more slice requests on the same network.

Time and number of use cases: despite the problem being NP-hard, the solver provided a solution in the majority of the instances studied within 10 minutes. Nevertheless, such a time limit affects the ability of the solver to provide optimal solutions to feasible problem instances. Understanding the impact of such a time limit on the output of the solver requires further investigation. We are also planning to explore the impact of μslicing as the number of use cases increases.

Extensions: μslicing and VNF co-location is a combination that can be used for several purposes. While we adopt it to save bandwidth, a mobile operator can use it to reduce energy consumption by minimizing the number of computing nodes that are active. We are planning to extend our model to support multiple objectives, which are modeled in the objective function through different multiplying factors of the decision variables. Mobile operators can benefit from such a model exploring different combinations of the multiplying factors and choosing the one that is more suitable depending on the context. Furthermore, given the performance issues of co-locating VNFs that compete for the same resources [38], we are also planning to include additional constraints on the type of VNFs that can be co-located in the same node.

ACKNOWLEDGEMENT

This work was supported by Business Finland - 5G FORCE.

REFERENCES

[1] 3GPP, "Service requirements for next generation new services and markets," 3rd Generation Partnership Project (3GPP), TS 22.261, Jan. 2018.

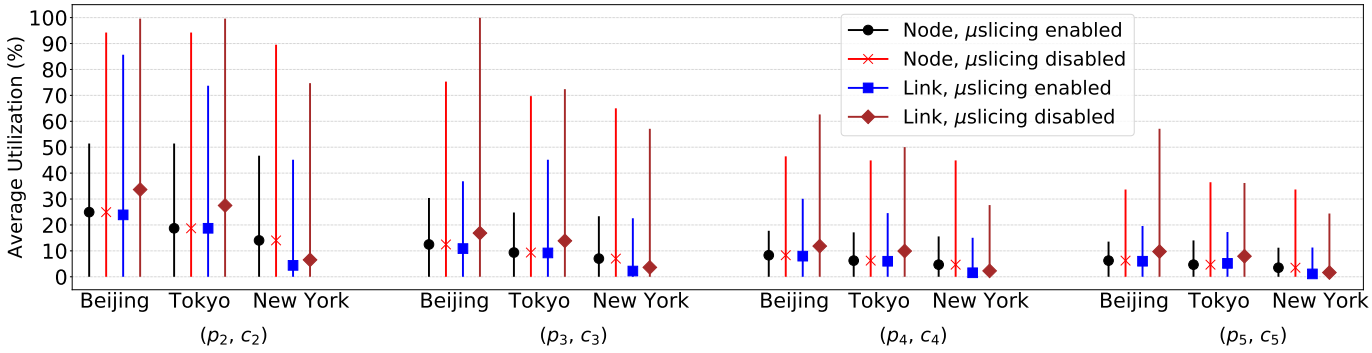

Fig. 3. **Node and link utilizations.** *The points indicate the average node and link utilizations while the error bars indicate the corresponding minimum and maximum values across the solved problem instances for a given configuration. Enabling μslicing results in a less skewed distribution of node and link utilization.*

[2] ——, "System architecture for the 5G System (5GS)," 3rd Generation Partnership Project (3GPP), TS 23.501, Dec. 2018.

[3] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A resource allocation framework for network slicing," in *IEEE INFOCOM*, 2018, pp. 2177–2185.

[4] A. Basta, A. Blenk, K. Hoffmann, H. J. Morper, M. Hoffmann, and W. Kellerer, "Towards a Cost Optimal Design for a 5G Mobile Core Network Based on SDN and NFV," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1061–1075, 2017.

[5] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Combined Virtual Mobile Core Network Function Placement and Topology Optimization with Latency Bounds," in *EWSDN*, 2015, pp. 97–102.

[6] ——, "Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization," in *IEEE NetSoft*, 2015, pp. 1–9.

[7] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan, "Multi-objective scheduling of micro-services for optimal service function chains," in *IEEE ICC*, 2017, pp. 1–6.

[8] F. Carpio, S. Dhahri, and A. Jukan, "VNF placement with replication for Load balancing in NFV networks," in *IEEE ICC*, 2017, pp. 1–6.

[9] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An Approach for Service Function Chain Routing and Virtual Function Network Instance Migration in Network Function Virtualization Architectures," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2008–2025, 2017.

[10] T. Kuo, B. Liou, K. C. Lin, and M. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *IEEE INFOCOM*, 2016, pp. 1–9.

[11] J. Kuo, S. Shen, H. Kang, D. Yang, M. Tsai, and W. Chen, "Service chain embedding with maximum flow in software defined network and application to the next-generation cellular network architecture," in *IEEE INFOCOM*, 2017, pp. 1–9.

[12] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On Dynamic Service Function Chain Deployment and Readjustment," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 543–553, 2017.

[13] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi, "Latency-aware composition of Virtual Functions in 5G," in *IEEE NetSoft*, 2015, pp. 1–6.

[14] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *IEEE CloudNet*, 2014, pp. 7–13.

[15] Z. Meng, J. Bi, H. Wang, C. Sun, and H. Hu, "CoCo: Compact and Optimized Consolidation of Modularized Service Function Chains in NFV," in *2018 IEEE ICC*, 2018.

[16] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker, "E2: a framework for NFV applications," in *SOSP*, 2015, pp. 121–136.

[17] A. Patel, M. Vutukuru, and D. Krishnaswamy, "Mobility-aware VNF placement in the LTE EPC," in *IEEE NFV-SDN*, 2017, pp. 1–7.

[18] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and Energy-efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach," *IEEE Transactions on Services Computing*, pp. 1–1, 2017.

[19] X. Song, X. Zhang, S. Yu, S. Jiao, and Z. Xu, "Resource-Efficient Virtual Network Function Placement in Operator Networks," in *IEEE GLOBECOM*, 2017, pp. 1–7.

[20] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware Virtual Network Function placement for Virtual 5G Network Infrastructure," in *IEEE ICC*, 2015, pp. 3879–3884.

[21] T. Wen, H. Yu, G. Sun, and L. Liu, "Network function consolidation in service function chaining orchestration," in *IEEE ICC*, 2016, pp. 1–6.

[22] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Network*, vol. 30, no. 3, pp. 81–87, 2016.

[23] 3GPP, "Procedures for the 5G System," 3rd Generation Partnership Project (3GPP), TS 23.502, Dec. 2017.

[24] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[25] J. G. Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[26] GSMA, "An Introduction to Network Slicing," https://www.gsma.com/futurenetworks/wp-content/uploads/2017/11/GSMA-An-Introduction-to-Network-Slicing.pdf, 2017.

[27] "SNDlib - library of test instances for Survivable fixed telecommunication Network Design," http://sndlib.zib.de/home.action, 2006.

[28] "The Internet Topology Zoo ," http://www.topology-zoo.org/.

[29] M. Jaber, M. A. Imran, R. Tafazolli, and A. Tukmanov, "5g backhaul challenges and emerging research directions: A survey," *IEEE Access*, vol. 4, pp. 1743–1766, 2016.

[30] "SPEC - Standard Performance Evaluation Corporation - SPEC virt_sc ® 2013," https://www.spec.org/virt_sc2013/.

[31] 3GPP, "General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access," 3rd Generation Partnership Project (3GPP), TS 23.401, Dec. 2017.

[32] 5GPPP, "5G Trials Roadmap," https://5g-ppp.eu/wp-content/uploads/2018/11/5GInfraPPP_TrialsWG_Roadmap_Version4.0.pdf.

[33] E. M. Metsälä and J. Salmelin, *LTE Backhaul - Planning and Optimization*. Wiley, 2015.

[34] S. Tabbane, "Core network and transmission dimensioning," https://www.itu.int/en/ITU-D/Regional-Presence/AsiaPacific/SiteAssets/Pages/Events/2016/Aug-WBB-Iran/Wirelessbroadband/core%20network%20dimensioning.pdf, 2016.

[35] Huawei, "eLTE Broadband Access Solution - QoS Technical Guide," 2014.

[36] 3GPP, "Policy and Charging Control Architecture," 3rd Generation Partnership Project (3GPP), TS 23.203, Sep. 2017.

[37] Z. Savic, "Lte design and deployment strategies," https://www.cisco.com/c/dam/global/en_ae/assets/expo2011/saudiarabia/pdfs/lte-design-and-deployment-strategies-zeljko-savic.pdf, 2011.

[38] C. Zeng, F. Liu, S. Chen, W. Jiang, and M. Li, "Demystifying the performance interference of co-located virtual network functions," in *IEEE INFOCOM*, 2018, pp. 765–773.